

(18)



Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number:

**0 023 568
B1**

(12)

EUROPEAN PATENT SPECIFICATION

(45) Date of publication of patent specification: **21.03.84**

(51) Int. Cl.³: **G 06 F 13/00, G 06 F 3/04**

(21) Application number: **80103555.1**

(22) Date of filing: **24.06.80**

(54) **Data interface mechanism for interfacing bit-parallel data buses of different bit width.**

(30) Priority: **30.07.79 US 62261**

(43) Date of publication of application:
11.02.81 Bulletin 81/6

(45) Publication of the grant of the patent:
21.03.84 Bulletin 84/12

(84) Designated Contracting States:
BE CH DE FR GB LI NL SE

(56) References cited:
FR - A - 2 373 831
US - A - 4 131 940

NACHRICHTENTECHNIK-ELEKTRONIK, Vol. 29,
no. 7, 1979, Berlin, DD TSCHLEBIEV et al.:
"Speicherkopplung üngleicher
Mikroprozessoren", pages 271-276
ELECTRONICS, vol. 53, no. 3, January 31, 1980,
New York, US WISTED et al.: "Peripheral
controller turns to I2L", pages 93-97

(73) Proprietor: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

(72) Inventor: **Dinwiddie, John Monroe, Jr.**
16931 Mead Hill Drive
Loxahatchee Florida 33470 (US)

(74) Representative: **Bonneau, Gérard**
Compagnie IBM France Département de Propriété Industrielle
F-06610 La Gaude (FR)

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European patent convention).

Courier Press, Leamington Spa, England.

EP 0 023 568 B1

Data interface mechanism for interfacing bit-parallel data buses of different bit width

Technical field to which the invention relates

This invention relates to data interface mechanisms for use in digital data processing systems for interfacing I/O units to a host processor wherein the I/O unit data bus is of a different bit width than the host processor channel data bus.

Relevant background art

It has been heretofore proposed to connect a wider data bus to a narrower data bus generally by means of a data register or memory having a width corresponding to that of the wider data bus. Selector circuitry is then used for connecting different portions of the data register or memory to the narrower data bus one at a time in an appropriate sequence.

Thus the article "Speicherkopplung ungleicher Mikroprozessoren" by Tschelebiev et al published in "Nachrichtentechnik-Electronik", Vol. 29, No. 7, 1979 describes a system in which a 16-bit microprocessor is coupled with a 8-bit microprocessor via a common storage in which two segments are simultaneously accessed by the wider microprocessor and only one segment at a time is accessed by the narrow microprocessor. But the drawback is that each microprocessor has to do its own addressing of the two storage segments.

In IBM System/360, Model 85, a 16-byte instruction register can be gated in from a 16-byte wide buffer data bus. A data path from this instruction buffer exists that can gate out four bytes starting with any even byte. This interface mechanism is described in IBM publication entitled: "IBM System/360 Model 85 — Functional Characteristics" p. 10 — Form A22-6916-1.

A similar problem is solved in patent FR—A—3,373,831 which describes a system in which memories connected between a processor and a group of buses, are used either for access n words at a time from the group of buses in parallel, or for word sequential access, a word at a time from each of the buses of the group.

While providing satisfactory operation in some applications, the interfacing mechanisms described in the above prior art can present a bottleneck to the movement of data in other applications. In particular, where multiple data handling units are connected to one side of the interface mechanism and the data handling units on opposite sides of the interface mechanism are not always ready to do a data transfer at the same moments, then delays can be encountered where one of the multiple units has to wait on the completion of a data transfer for another of the multiple units.

When an I/O controller is used to couple multiple I/O devices to the I/O channel bus of a host processor, the data buses which couple the

I/O units to the I/O controller have for example a width of one byte and the I/O channel data bus has a multiple byte width. In this case, the host processor might transfer a multi-byte data word to an interface mechanism data register in the I/O controller for subsequent retransfer to a first I/O unit A. If the I/O unit A should not be ready to receive such data, then the data must remain in the data register. In such case, a second I/O unit B, which is ready to transfer data to the host processor, would have to sit and wait until the data register in the interface mechanism became available. Thus, for the case of multiple I/O units, the use of such a data register for interface purposes would represent a definite bottleneck in the system.

This bottleneck problem could be alleviated to some extent by providing separate data registers for each of the different I/O units. This, however, would increase the circuit complexity and would require further circuitry for determining which of the data registers should be connected to the I/O channel bus at any given moment. The bottleneck problem might also be alleviated to some extent by using some further form of storage mechanism for immediately removing the multi-byte data word from the interface mechanism data register after it is received from the host processor or, conversely, for not transferring any data from the I/O unit to the interface mechanism data register until a complete multibyte word has been assembled. This however, would require additional circuitry and would, in general, tend to increase the number of steps involved in the overall transfer of data from the host processor to an I/O unit, or vice versa.

Disclosure of the invention

There is described herein a new and improved data interface mechanism for interfacing a M-byte data bus connected to a host processor with an N-byte data bus where the ratio of M to N is an integer greater than 1 which N-byte data bus is connected to I/O devices by the intermediary of a microprocessor. In the embodiment illustrated herein, this improved interface mechanism provides an automatic and highly efficient mechanism for converting data bytes into data words, and vice versa.

The mechanism of the invention, like the system disclosed in Tschelebiev's article, comprises a number of random access storage units, the number of such storage units being equal to the ratio of M to N and each such storage unit having a width of N bytes; a common addressing circuitry connected to the microprocessor's address bus to receive first address bits and to supply the first address bits to all of the random access storage units; a logic circuitry which is connected to receive a

set of first control signals (CS1, CS2) and a further control signal (CS3) for activating a different one of the random access storage units in response to each of the first control signals, and for activating simultaneously all of the random access storage units in response to the further control signal, thereby enabling a storage operation to be carried out in the activated random access storage unit or units at the address specified by the first address bits; a first data transfer mechanism which is connected to receive the set of first control signals and which, in response to each of the first control signals, couples the random access storage unit currently activated by the logic circuitry to the N-byte data bus for transferring an N-byte data segment between the N-byte data bus and the activated random access storage unit; a second data transfer mechanism which is connected to receive the further control signal and which, in response thereto, couples simultaneously all of the random access storage units to the M-byte data bus for transferring in parallel an M-byte data segment between the M-byte data bus and the random access storage units; and a control signal generating means for generating the set of first control signals and the further control signal, the first control signals being generated in an order such that the random access storage units are accessed one at a time in a rotating manner by the logic circuitry.

The mechanism of the invention is characterised in that the first address bits form the lower order address bits of an address on the microprocessor's address bus; the control signal generating means comprises decoder circuitry which is connected to receive the higher order address bits of an address on the microprocessor's address bus and which, in response to the higher order address bits of an address in one of a set of first address ranges, generates a respective one of the first control signals and, in response to the higher order address bits, of an address in a further address range, generates the further control signal; and the microprocessor is provided with means for placing on the microprocessor's address bus a series of first addresses, each of which is an address in one of the set of first address ranges, to cause the transfer of data in N-byte segments between the random access storage elements and the N-byte data bus, and for placing on the microprocessor's address bus, an address in the further address range to cause the transfer in parallel of M bytes of data between the random access storage units and the M-byte data bus.

Advantageous effects of the invention

This invention reduces the overall time required to move data between multiple I/O devices and a host processor. It also provides considerable flexibility as to the manner in which the data is transferred. The data can be transferred on to the host processor as soon as

a word's worth of bytes are assembled in the storage units, or alternatively, a desired number of words can be accumulated in the storage units and then transferred to the host processor in a cycle steal mode, the data transfer to or from the processor main storage unit taking place during a stolen processor clock interval.

For a better understanding of the present invention reference is made to the following description taken in connection with the accompanying drawings.

Brief description of the drawings

Referring to the drawings:

Figure 1 is a schematic block diagram of a digital data processing system showing the incorporation therein of a mechanism constructed in accordance with the present invention.

Figures 2A, 2B, 2C and 2D form a single figure (Figure 2) which shows in greater detail the construction of the mechanism of Figure 1.

Figures 3—6 show various command, control block and status word formats and the like used in the Figure 1 data processing system.

Figure 7 is a timing diagram showing what happens for a more or less typical instruction cycle for the microprocessor used in the mechanism shown in Figure 2.

Figure 8 shows in greater detail the internal construction of a typical plural channel direct memory access (DMA) controller which can be used in the mechanism shown in Figure 2.

Figure 9 is a timing diagram showing the various signal waveforms for two typical successive DMA cycles of the DMA controller of Figure 8.

Figures 10A and 10B show in greater detail the internal construction of the interrupt and cycle steal hand-shaking unit of Figure 2C.

Figures 11A and 11B show in greater detail the internal construction of the storage control logic of Figure 2A and also show the logic for developing the direction (D) and output enable (EO) signals for the two-way drivers which are used to transfer data between the host processor channel bus and the I/O controller storage unit.

Figure 12 is a storage address range map of the lower portion of the total address range capable of being addressed by the microprocessor of Figure 2A.

Figure 13 is an enlargement of a portion of the Figure 12 address range map, the part shown in Figure 13 being applicable to the two IDCB address range segments shown in Figure 12.

Figure 14 shows in greater detail the internal construction of the host DMA request logic of Figure 2B and also the logic for developing the strobe and output enable control signals for the data register of Figure 2A.

Figure 15 shows a modification for the Figure 2 I/O controller whereby the data trans-

fers between one of the I/O units and the controller storage unit are also handled by the channel DMA controller of Figure 2B.

Figure 16 shows a further modification of the Figure 2 controller wherein a pair of DMA controllers are used to control the data transfers between three different I/O units and the controller storage unit, and Figure 17 shows in greater detail the construction of the DMA contention logic of Figure 16.

Description of one embodiment

Unless otherwise indicated by the context, the term "data" is used herein in its broadest sense as including any kind of information such as alphanumeric data, status information, control information, address values and the like.

For the present embodiment, a "word" is assumed to be composed of two bytes and a "byte" is assumed to be composed of eight bits.

Referring to Figure 1, there is shown a block diagram of a digital data processing system which includes an input/output (I/O) controller. The system includes a host processor 1, an I/O controller 2 and a plurality of I/O units 3—6. Coupled to the host processor 1 is a host processor main storage unit 7. The host processor 1 is constructed to communicate with various I/O controllers and I/O units by means of a host processor I/O channel bus 8. This channel bus 8 is connected to and driven by a channel portion 9 of the host processor 1. Channel portion 9 is also directly connected to the main storage unit 7 by way of a channel storage bus 10. This channel storage bus 10 enables the cycle stealing of data between the channel bus 8 and the main storage unit 7 without interrupting the program being executed in the host processor 1.

For sake of example herein, the bus 8 may be assumed to be a channel bus similar to that described in United States Patent 4,038,642, entitled "Input/Output Interface Logic For Concurrent Operations" and issued on July 26, 1977, to Messrs. Boucknecht et al. This patent also described an I/O controller and its manner of use in connecting I/O units to the host processor channel bus 8.

The I/O controller 2 is a representative embodiment of a new and improved I/O controller. It provides various advantages and improvements over and above those which are provided by the existing controllers.

The new and improved I/O controller 2 is a microprocessor based I/O controller and includes a microprocessor 11, a control program storage unit 12, a direct memory access (DMA) controller unit 13, a programmable interrupt controller (PIC) unit 14 and, optionally, a user storage unit 15. Typically, each of these units 11—15 is comprised of one or more integrated circuit chips and each of these units 11—15 is coupled to a microprocessor I/O bus 16 which is of the proper construction as required by the micro-

processor 11. The I/O units 3—6 are also coupled to the microprocessor I/O bus 16 by way of their respective ones of device control units 17—20.

For sake of example herein, the microprocessor 11 is assumed to be a single chip 8-bit microprocessor. The control program storage unit 12 is assumed to be a read only storage unit and is constructed to contain the various operating instructions and operating routines which are used by the microprocessor 11 in supervising and controlling the data transfer activities in the I/O controller 2.

For sake of example, the direct memory access controller 13 is assumed to be a single chip four-channel DMA controller and the programmable interrupt controller 14 is assumed to be a single chip eight-request interrupt controller.

A primary feature of the new and improved I/O controller 2 is the use of a dual port random access storage mechanism 22 to provide the data transfer interface between the microprocessor bus 16 and the host processor channel bus 8. One port of this storage mechanism 22 is coupled to the host processor channel bus 8 and the other port is coupled to the microprocessor bus 16. Data passing from the channel bus 8 to the microprocessor bus 16 or vice versa is at least temporarily stored in this storage mechanism 22.

The construction of the I/O controller 2 is such that the microprocessor 11 thinks that this dual port storage unit 22 is its own private random access storage unit. In particular, the storage unit 22 is coupled to the microprocessor 11 in generally the same manner as any other random access storage unit is normally coupled to a microprocessor. Thus, microprocessor 11 can transfer data into or out of the storage unit 22 in its normal manner. Nevertheless, the dual port storage unit 22 can be accessed directly by the host processor 1 and the host processor 1 can transfer data to or from the storage unit 22 during such access. The construction is, however, such that this direct accessing by the host processor 1 is transparent to the microprocessor 11. Thus, the dual port storage mechanism 22 is operated in the manner of a storage unit which is shared by both the microprocessor 11 and the host processor 1, with the host processor accesses being transparent to the microprocessor 11.

The host processor 1 can cause the initiation or termination of I/O operations in the I/O controller 2 by sending thereto via channel bus 8 immediate device control blocks (IDCB's) which include a one-byte I/O command and a one-byte device address. In the present embodiment, these two items, the I/O command and the device address, are sent out over the address bus portion of the channel bus 8. Each of the I/O units 3—6 is assigned its own unique device address. An address decoder 23 monitors the channel bus 8. When it detects the

occurrence of the unique device address for one of the I/O units 3—6, it generates an address gate capture signal on its output line 24. This signal is supplied to a four-byte command register file 25 to cause the storage therein of the one-byte I/O command then appearing on the channel bus 8. The address decoder 23 also activates the appropriate one of the device select lines in a four-line device select bus 26, these four device select lines running to four different interrupt request inputs of the programmable interrupt controller 14. In due course, the interrupt controller 14 sends an interrupt request for the selected I/O unit to the microprocessor 11 via the microprocessor bus 16. In due course, the microprocessor 11 recognizes this interrupt request and fetches the corresponding I/O command from the register file 25. In due course, the microprocessor 11 initiates whatever action is called for by this I/O command.

The host processor I/O command mode is also used for transferring data between the host processor 1 and the dual port storage unit 22. More particularly, during the occurrence of the I/O command and device address on the channel bus 8, a word (two-bytes) of data can also be placed on the data bus portion of the channel bus 8 by either the host processor 1 or the dual port storage unit 22. For the case of a write type I/O command, the host processor 1 places the data word on the channel bus, after which such data word can be written into the dual port storage 22. Conversely, for the case of a read type I/O command, the dual port storage 22 places a data word on the channel bus 8 and such data word is ready into the host processor 1. In either case, the addressing of the dual port storage 22 is accomplished by some of the I/O command and device address bits appearing on the address bus portion of the channel bus 8. In other words, the host processor 1 supplies the address bit values which are used to address the dual port storage 22. The construction of the I/O controller 2 is such that this accessing of the dual port storage 22 does not interrupt the program being executed by the microprocessor 11 and, hence, this host processor accessing is transparent to the microprocessor 11.

A second and different mode of data transfer between the host processor 1 and the I/O controller 2, or, more particularly, the controller dual port storage unit 22, is also provided. This second mode is a so-called cycle steal data transfer mode with the individual data word transfers being initiated and controlled by the I/O controller 2 and with the data words being transferred into or out of the host processor main storage unit 7 during storage cycles stolen from the host processor. Normally, this cycle steal mode is used for transferring a multi-word block of data between the host processor 1 and the controller storage unit 22. In the present embodiment, these cycle steal data transfers are supervised and controlled by the direct

memory access controller 13. As such, for each data word transfer, the DMA controller 13 supplies a first address to the host processor 1 by way of a cycle steal address register 27 and a second address to the dual port storage unit 22 by way of the microprocessor bus 16. The address supplied to the host processor 1 selects the location in main storage 7 to (or from) which the data is to be transferred and the address supplied to the dual port storage 22 selects the location in the storage 22 from (or to) which the data is to be transferred. The data word is moved into (or out of) the host processor main storage unit 7 in a cycle steal mode (via storage bus 10), which means that such data transfer does not cause an interruption of the program being executed by the host processor 1. This cycle steal data transfer mode requires the use of two channels in the DMA controller 13, one for supplying the host processor main storage addresses to the cycle steal address register 27 and the other for supplying the controller storage addresses to the storage unit 22. The DMA controller 13 also includes a word counter for keeping track of how many words in a multi-word block of data remain to be transferred.

Operation of the DMA controller 13 is controlled by host DMA request logic 28. For any given multi-word cycle steal transfer operation, the address counters and the word counter in the DMA controller 13 are initially loaded to the proper starting values by the microprocessor 11. Then the microprocessor 11 issues appropriate "start" signals to the request logic 28 via line 29. Thereafter, the DMA controller 13 and the request logic 28 take over to handle the cycle stealing of the multi-word block of data. For each word transfer, the request logic 28 sends to a handshake, interrupt and miscellaneous controls unit 30 a cycle steal request signal via line 31. In response thereto, the controls unit 30 sends a cycle steal request signal to the host processor 1. When the host processor channel portion 9 is ready to do the data word transfer, it sends back a service gate signal to the control unit 30 which, in response thereto, produces a service gate capture signal which is supplied via line 32 to the request logic 28. This signal is used to coordinate the operation of the DMA controller 13 with the operation of the host processor channel portion 9. The handshake, interrupt and miscellaneous controls 30 are generally similar to those described in the above-cited U.S. Patent 4,038,642 to Bouknecht et al and, hence, will not be described in detail herein.

As will be seen, the I/O controller 2 includes circuitry for interleaving the host processor I/O command type data transfers with the individual data word cycle steal transfers provided by the DMA controller 13. Thus, two different modes of data transfer are provided between the host processor 1 and the I/O controller 2, with the individual transfers for the

two modes being interleaved to provide a minimum of delay and a minimum of interference with one another.

A third mode of data transfer remains to be considered, namely, the mode or manner of transferring data between the dual port storage unit 22 and the I/O units or I/O devices 3—6. In the present embodiment, the microprocessor 11 is used to handle and control this transfer of data between the storage unit 22 and the I/O devices 3—6. For the case of a storage unit to device transfer, a first microprocessor instruction cycle is used to address the storage unit 22 and to transfer a byte of data from the storage unit 22 to an internal register in the microprocessor 11. A second microprocessor instruction cycle is then used to address the desired I/O device and to move the data byte from the microprocessor internal register to the selected I/O device. When transferring data in the opposite direction, this sequence is performed in reverse, namely, the microprocessor 11 fetches a byte of data from a particular I/O device and then, during its next instruction cycle, writes such byte of data into the storage unit 22.

The device control units 17—20 for the respective I/O devices 3—6 inform the microprocessor 11 as to when they are ready to send to (or receive from) the storage unit 22 a byte of data. This informing action is accomplished by way of interrupt request signals which are sent to the programmable interrupt controller 14. For each such interrupt request, the programmable interrupt controller, in turn, sends an interrupt request to the microprocessor 11 by way of the microprocessor bus 16. The programmable interrupt controller 14 includes a priority resolver which operates when plural requests are received to determine the order in which these requests are passed on to the microprocessor 11.

In addition to performing its I/O data transfer activities, the microprocessor 11 can also be used to offload some of the programming functions normally performed in the host processor 1. The microprocessor 11 can, for example, do some of the number crunching normally done in the host processor 1. There are several ways of doing this. A typical way would be to have the host processor 1 transfer the appropriate number crunching program routine into the dual port storage unit 22. The routine transferred to the dual port storage 22 would include all of the instructions needed by the microprocessor 11 for subsequently receiving from the host processor 1 the numbers to be crunched, doing the number crunching and thereafter transferring the results back to the host processor 1. After the program routine has been transferred, then whenever the host processor 1 has a set of the numbers to be crunched, it advises the microprocessor 11 of this fact (via a particular I/O command) and sends the numbers over to the dual port storage unit 22. Thereafter, as

time permits, the microprocessor 11 does the number crunching under the control of the number crunching program instructions previously stored in the dual port storage unit 22. After completion of the number crunching, microprocessor 11 causes the results to be sent back to the host processor 1. Typically, the number crunching program, the numbers to be crunched and the results, will be transferred between the host processor 1 and the dual port storage unit 22 by means of the host processor cycle stealing mode which is controlled by the DMA controller 13.

An interesting aspect of the host processor offloading capability of the I/O controller 2 is that the particular host processor functions offloaded to the I/O controller 2 can be changed from time to time, if desired. Thus, for example, during a first interval of time a first type of number crunching operation could be offloaded to the I/O controller 2 and later, during a second interval of time, a second and different type of number crunching function could be offloaded to the I/O controller 2 to replace the first number crunching function. Thus, where desired, the mission of the I/O controller 2 can be changed from time to time by the host processor 1.

Description of the embodiment of Figure 2

Referring now to Figures 2A, 2B, 2C and 2D, there is shown in greater detail the construction of the I/O controller 2 of Figure 1. Figure 2D should be placed on the right side of Figure 2C, Figure 2A above Figure 2C, Figure 2B above Figure 2D, to form a single figure which will be referred to herein as Figure 2. A minor drawing difference to be noted is that a unit 33 shown in Figure 2C is intended to include some, but not all, of the functions covered by the handshake, interrupt and miscellaneous controls unit 30 of Figure 1. More particularly, the unit 33 of Figure 2C includes only the interrupt and cycle steal handshaking functions of the controls unit 30 of Figure 1.

With reference to Figure 2, the host processor channel bus 8 is comprised of a two-byte data bus 34, a two-byte address bus 35 and a multi-line control bus 36.

The two-byte (or one word) data bus 34 includes 16 parallel data bit lines which are subgrouped into a first 8-bit data bus 34a for the high order byte of the 2-byte host processor data word and a second 8-bit data bus 34b for the low order byte of the 2-byte data word. In a similar manner, the address bus 35 is composed of 16 parallel address bit lines which are subgrouped into an 8-bit address bus 35a for the higher order address bits and an 8-bit address bus 35b for the lower order address bits. The control bus 36 is comprised of approximately 45 parallel control lines and these control lines and their functions are discussed in greater detail in the above-cited U.S. Patent 4,038,642 to Boucknecht et al.

The microprocessor I/O bus 16, on the other hand, is comprised of a 1-byte data bus 37, a 2-byte address bus 38 and a multi-line control bus 39. The data bus 37 has 8 parallel data bit lines. The address bus 38 has 16 parallel address bit lines and these are subgrouped into a first 8-bit address bus 38a for the higher order address bits and a second 8-bit address bus 38b for the lower order address bits. The control bus 39 includes somewhere on the order of 25 parallel control signal lines, corresponding approximately to the number of control terminals of the microprocessor 11 plus a few additional lines for the control terminals for some of the other units which do not match up with or complement the microprocessor control terminals.

As shown in Figure 2A, the dual port storage unit 22 is actually comprised of a number of separate random access storage units, the number of such storage units being equal to the ratio of M to N, where M denotes the width of the host processor data bus 34, and N denotes the width of the microprocessor data bus 37, with M being a multiple of N. In the present embodiment, M is equal to two bytes and N is equal to one byte, giving a ratio of M to N of 2. Thus, in the present embodiment, the dual port storage mechanism 22 is composed of two separate random access storage units 22a and 22b, each having a width of one byte. The storage unit 22a is used for storing the high order bytes of the various two-byte data words and thus is coupled to the high byte data bus 34a of the host processor channel bus 8. Storage unit 22b is used for storing the low order bytes of the various two-byte data words and, as such, is coupled to the low byte data bus 34b of the host processor channel bus 8.

For the more general case, the number of such separate storage units is made equal to the ratio of M to N. Thus, for example, for the case of a host processor having a 4-byte channel data bus and a microprocessor having a 1-byte data bus, four such separate storage units would be used. Also, in the more general case, the width of each such storage unit should be equal to N, the width of the narrower of the two data buses.

As indicated in Figure 2A, the I/O controller 2 also includes a separate and different selectively operable N-byte data transfer mechanism for each of the different storage units 22a and 22b. In the present embodiment, these data transfer mechanisms take the form of one-byte two-way drivers 40 and 41. For sake of example, each of these drivers 40 and 41 is assumed to be an 8-bit parallel bidirectional bus driver. One side of the 8-bit driver 40 is coupled by way of an 8-bit storage bus 42 to the data terminals of the high byte storage unit 22a and the other side or set of I/O terminals of the driver 40 is coupled to the 8-bit microprocessor data bus 37. In a corresponding manner, one side of the 8-bit driver 41 is

coupled by way of an 8-bit storage bus 43 to the data terminals of the low byte storage unit 22b and the other side of driver 41 is coupled to the 8-bit microprocessor data bus 37.

Each of the drivers 40 and 41 has two control terminals, namely, a direction control terminal D and an output enable control terminal OE. When the direction control terminal D is at the zero level, data can flow from right to left and, when D is at the one level, data can flow in the opposite direction, namely, from left to right. When the output enable terminal OE is at the zero level, all output lines of the drivers are set to a high impedance output state and the driver is disabled so that no data can pass therethrough. When the OE terminal is set to the one level, the driver is enabled and is in a condition to pass data in the direction determined by the binary level at the direction control terminal D.

As will be seen, these drivers 40 and 41 are rendered operative, that is, enabled, only when data is being transferred between the storage units 22a and 22b and the microprocessor data bus 37. Also they are enabled in an alternating manner for transferring successive data bytes between the microprocessor data bus 37 and alternate ones one of the storage units 22a and 22b. In other words, for a first data byte, driver 40 is turned on and driver 41 is turned off to enable this data byte to be transferred to or from the high byte storage unit 22a. Then, for the next data byte, driver 41 is turned on and driver 40 is turned off to enable the data byte to be transferred to or from the low byte storage unit 22b. In this manner, only one byte at a time is transferred to or from the one-byte microprocessor data bus 37.

The I/O controller 2 further includes a selectively operable M-byte data transfer mechanism for coupling the data terminals of the different storage units 22a and 22b to different N-byte subgroups of the M-byte host processor channel data bus. In the present embodiment, M is two and N is one so that the data terminals of the different storage units 22a and 22b are coupled to the respective ones of the one-byte subgroups 34a and 34b of the 2-byte channel data bus 34. This data transfer mechanism includes 16-bit two-way driver 44, 16-bit two-way driver 45, a 16-bit or 2-byte data register 46 and a register control unit 47. The connections are such that the high byte storage unit 22a is connected to the high byte data bus 34a and the low byte storage unit 22b is connected to the low byte data bus 34b.

These drivers 44 and 45 also have direction control terminals D and output enable control terminals OE. In this case, however, the orientation of the drivers 44 and 45 is such that when the direction control terminal D is at the zero level, the data flow direction is from left to right, that is, from the channel data bus 34 to the storage units 22a and 22b. For the moment, it is assumed that these direction and output

enable control signals are obtained from a storage control logic unit 48 and, in fact, their manner of generation will be explained hereinafter in connection with the details of the storage control logic 48.

For sake of example, the data register 46 is assumed to be a pair of units, each such unit including eight bipolar latches, each having a three-state output buffer. A strobe signal STB is used to load the latches and an output enable signal OE is used to enable the output buffers. When not enabled, these latch output buffers present a high output impedance to the register output terminals. In the present embodiment, the input terminals of the data register 46 are coupled to the same data lines as are the output terminals of the data register 46. Thus, each individual bit input terminal is, in effect, connected to its corresponding individual bit output terminal. As will be seen, there may be occasions when both the strobe and output enable control signals are at the one level at the same time. This means that the latch output buffers will be enabled at the same time that the latches are being loaded. This is a permissible condition and will not cause injury to either the latches or the output buffers.

As will be seen, the two-way driver 45 is never enabled at the same time that one or the other of the one-byte drivers 40 and 41 are enabled. Thus, the data terminals of the storage units 22a and 22b can be connected to one or the other of the channel data bus 34 and the microprocessor data bus 37, but never to both at the same time. As is apparent, the two-way driver 45 serves to transfer data to or from both of the storage units 22a and 22b in a simultaneous manner. This is in contrast to the alternate byte-by-byte transfers provided by the one-byte drivers 40 and 41.

The purpose of the data register 46 is to temporarily store a 2-byte data word for those cases where the host processor 1 is not ready to accept the data word at the same moment that the I/O controller is wanting to transmit the data word and vice versa. For example, the I/O controller 2 can read a data word out of the storage units 22a and 22b and, if the host processor 1 is not quite ready to receive it, then such data word is temporarily held in the data register 46. Thereafter, when the host processor 1 becomes ready, the two-way driver 44 is enabled to place the data word being held in the data register 46 onto the host processor data bus 34. If, on the other hand, the host processor 1 were ready at the same time as the I/O controller 2, then both the two-way driver 44 and the two-way driver 45 would be enabled at the same time so that the data word could be passed straight through to the host processor data bus 34. In this case, the presence of the data register 46 is of no consequence.

Similar considerations apply when the data word is being transferred in the opposite direction, namely, from the host processor data bus

34 to the storage units 22a and 22b. If the storage units 22a and 22b are not ready in time, then the data word is temporarily held in the data register 46 and the driver 45 is not enabled until the storage units 22a and 22b become ready. Thus, the use of the data register 46 helps take into account differences in timing between the host processor 1 and the I/O controller 2.

Before discussing the various addressing related mechanisms, it is helpful to consider the various items that may appear at different times on the host processor channel bus 8. These items are shown and explained in Figures 3—6. Figure 3 shows the layout of the immediate device control block (IDCB) which the host processor 1 places on the channel bus 8 when it wants to send an I/O command to a peripheral unit such as the I/O controller 2. This is a 4-byte control block wherein the first byte (Byte 0) is an 8-bit I/O command, the second byte (Byte 1) is an 8-bit device address and the third and fourth bytes (Bytes 2 and 3) contain either a 16-bit direct program control (DPC) data word or a 16-bit device control block (DCB) starting address. The I/O command (Byte 0) is sent out on the high order address byte bus 35a and the device address is sent out on the low order address byte bus 35b. The high order and low order bytes of the 2-byte data word or 2-byte DCB address are sent out on the respective ones of the high order data byte bus 34a and the low order data byte bus 34b, the high order byte being IDCB Byte 2 (bits 16—23) and the low order byte being IDCB Byte 3 (bits 24—31). All four of these IDCB bytes are sent out simultaneously.

The chart of Figure 6 explains the significance of the different classes or categories of I/O commands.

The abbreviations used in Figure 6 are as follows:

Term	Meaning
RD	Read
WR	Write
CONTR	Control
ST	Start
STS	Status
STE	Steal
CYC	Cycle
CHAN	Channel
H	Halt

As is apparent, bit 1 of the I/O command is used to distinguish between read (RD) type and write (WR) type operations. Read operations are those where data or other information is to be transferred from an I/O unit to the host processor and, conversely, write operations are those where data or other information is transferred from the host processor to an I/O unit.

Another way of classifying the I/O operations is as to whether they are direct program

control (DPC) operations or cycle steal operations. Considering first the case of DPC operations, each DPC read type command enables a 2-byte word of data or status information to be transferred from the I/O controller 2 to the host processor 1. Each DPC write type operation enables a 2-byte word of data or control information to be transferred from the host processor 1 to the I/O controller 2. The DPC data word (IDCB bits 16—31) is transferred by way of the channel data bus 34 and is stored into or transferred out of the dual port storage unit 22a, 22b, with the higher order byte (Byte 2 or bits 16—23) being stored into or read from the high byte storage unit 22a and the lower order byte (Byte 3 or bits 24—31) being stored in or read from the low byte storage unit 22b. This type of data transfer is called "DPC" because the transfer of each data word is under the direct control of the host processor program and the host processor must issue a separate I/O command for each word transferred.

As previously mentioned, the actual cycle stealing of data into or out of the host processor main storage unit 7 is controlled by the I/O controller 2. Before such cycle steal operations can be commenced, however, it is necessary for the host processor 1 to send a start cycle steal command to the I/O controller 2. Bytes 2 and 3 (bits 16—31) of the IDCB for such a start cycle steal (ST CYC STE) command contain the address in main storage 7 at which is stored the first word (Word 0) of an eight-word device control block (DCB). A typical format for this eight-word device control block is as follows:

Word 0	Control word
Word 1	Device parameter word 1
Word 2	Device parameter word 2
Word 3	Device parameter word 3
Word 4	Residual status block address
Word 5	Next DCB address
Word 6	Byte count
Word 7	Main storage data address

The main storage starting address (Word 0 address) of this device control block is transferred by way of the channel data bus 34 and stored into the dual port storage units 22a and 22b. This main storage starting address is thereafter used by the I/O controller 2 to cycle steal the eight words of the device control block out of the main storage unit 7, such DCB words being transferred to and stored into the dual port storage units 22a and 22b.

The microprocessor 11 thereafter uses the information contained in some of these DCB words to initialize the DMA controller 13 to the proper starting conditions for the desired data transfer cycle stealing operations. Thereafter the cycle stealing of the individual data words into or out of the main storage unit 7 is controlled by the DMA controller 13. For each of these individual data word transfers, the usage

of the host processor channel bus 8 is as indicated in Figure 4. The main storage data address (Bytes 0 and 1) is sent from the I/O controller 2 to the host processor 1 via the channel address bus 35 and the data word (Bytes 2 and 3) to be transferred is transferred via the channel data bus 34. The main storage address is the address in the main storage unit 7 to which or from which the data word is to be transferred.

A further type of data word that is at times transferred by way of the channel data bus 34 is the interrupt identification (ID) word shown in Figure 5. This interrupt ID word is sent from the I/O controller 2 to the host processor 1 for purposes of notifying the host processor 1 of some condition or event that has occurred out in the I/O controller 2 or to notify the host processor 1 that the I/O controller 2 needs service or, more accurately, that the I/O unit identified by the device address needs service. The interrupt information byte (IIB) identifies the type of service that is needed. As will be seen, the proper interrupt ID word is set into the storage units 22a and 22b by the microprocessor 11 and is thereafter transferred from such storage units 22a and 22b to the host processor 1 by way of the channel data bus 34.

As indicated in Figure 2A, the address decoder 23 includes an 8-bit address comparator circuit 50. One set of input terminals for this comparator 50 are coupled by way of the two-way driver 51 to the low order address byte bus 35b for receiving the device address byte portion of an IDCB. The other set of input terminals for the comparator 50 are coupled to address jumpers 52 which are jumpered to represent the predetermined or preassigned device addresses which are assigned to the I/O units which are attached to the I/O controller 2. When the device address appearing on the low byte address bus 35b matches one of the device addresses provided by jumpers 52, the address comparator 50 produces a "controller select" signal on an output line 53. If, at this time, the host processor 1 is sending out an address gate signal on the address gate line of the channel control bus 36 (which will be the case if a valid IDCB is present on the channel bus 8), then an AND circuit 54 will produce an address gate capture signal on its output line 24. Among other things, this address gate capture signal is supplied by way of a driver circuit 55 to produce an address gate return signal on the address gate return line of the channel control bus 36. This tells the host processor 1 that the device address has been properly detected and that the I/O controller 2 is ready to proceed with the IDCB data transfer.

For sake of example, it is assumed that the two-way driver 51 is comprised of two 8-bit parallel bidirectional driver units. For simplicity, they are shown as a single block in Figure 2A. One of these 8-bit drivers connects the high byte channel address bus 35a to an internal

high byte address bus 56, while the other of these 8-bit drivers connects the low byte channel address bus 35b to an internal low byte address bus 57. The orientation of the driver 51 is such that, when the direction control signal D is at the zero level the driver 51 is set to transfer data from left to right or, in other words, from the channel address bus 35 to the internal buses 56 and 57. The output enable control terminal OE (not shown) of driver 51 is permanently connected to a voltage source so that the outputs of the driver 51 are always enabled. Because of this and because the direction control signal is normally at the zero level to provide a left to right data transfer direction, the address comparator 50 is able to monitor the low byte address bus 35b on an almost continuous basis. The only time it cannot is when a main storage address is being sent to the host processor 1 by the cycle steal address register 27. During such a main storage address transfer, a cycle steal service gate (CS/SG) capture signal is present to place the direction control terminal D at the one level to cause the direction of data transfer to be from right to left for the duration of such signal.

The cycle steal address register 27 is a 16-bit register and, for example, may be comprised of a pair of 8-bit input/output port units. The main storage address to be sent to the host processor 1 is obtained from the DMA controller 13 via microprocessor address buses 38a and 38b and is strobed into the address register 27 by the output signal from an AND circuit 58. This strobe signal is produced when the AND circuit 58 receives both a MEMW (memory write) signal from the MEMW line of the microprocessor control bus 39 and a DACK 0 signal from the DMA controller 13. These signals will be discussed in greater detail hereinafter. The output enable signal for the address register 27 is provided by the same service gate capture signal as was discussed for the driver 51.

When needed, the device address value provided by the address jumpers 52 can be supplied by way of 8-bit drivers 59 to the microprocessor data bus 37. More particularly, the microprocessor 11 can transfer this address jumper address value to the low byte storage unit 22b for purposes of providing the device address portion of the interrupt ID word shown in Figure 5.

The appearance of an address gate capture signal on line 24 causes several things to happen. For one thing, it enables a device select decoder 60 to decode the device address appearing on the internal low byte address bus 57 and to activate the particular one of its device select output lines 26 which corresponds to that device address. Thus, for example, if the device address on the internal bus 57 is for I/O device A, then the device A (DEV. A) device select line is energized. As previously indicated, these device select lines 26 run to the programmable interrupt controller (PIC) 14

as is better shown in Figure 2D. These device select lines A—D are connected to four different interrupt request inputs of the interrupt controller 14.

When one of these device select lines is activated, it causes the interrupt controller 14 to send an interrupt request signal to the microprocessor 11 by way of the microprocessor control bus 39. After acknowledgment of the interrupt request by the microprocessor 11 (via an interrupt acknowledgment signal on another line of the control bus 39), the interrupt controller 14 will send to the microprocessor 11 a CALL instruction which causes the microprocessor 11 to branch to the appropriate service routine for processing the I/O command for the I/O device to which such command is directed. A separate command processing service routine is provided in the control program storage 12 for each of the different I/O units or I/O devices attached to the I/O controller 2.

At this point it should be noted that the I/O controller of Figure 2 uses a pair of programmable interrupt controllers as opposed to the single interrupt controller 14 shown in Figure 1. The second interrupt controller is identified by reference numeral 61 in Figure 2D and is connected in cascade with the first interrupt controller 14 to provide, in effect, a single interrupt controller capable of handling twice as many interrupt requests.

A second result which is produced by the appearance of an address gate capture signal on line 24 is that the I/O command appearing on internal address bus 56 is stored into the command register file 25. For sake of example, it is assumed that the command register file 25 is comprised of two 4-by-4 register files. These two 4-bit wide register files are operated in unison to provide, in effect, a single register file having a width of 8 bits or 1 byte, with the four one-byte locations being separately addressable. In other words, register file 25 is just like a stack of four addressable 1-byte registers. The construction of the register file 25 is such as to permit simultaneous writing into one of the byte locations and reading from another of the byte locations.

The address gate capture signal on line 24 is supplied to the write enable terminal of the register file 25 and the lowest order two address bits on the low order internal address bus 57 are supplied to the write select or write addressing terminals of the register file 25. In the present embodiment, the four I/O devices attached to the I/O controller 2 are assigned four consecutive device addresses. In this case, the two lowest order device address bits appearing on bus 57 are sufficient to distinguish between the four I/O device addresses. Thus, a different one of the 1-byte locations in the register file 25 is assigned to each of the different I/O devices attached to the I/O controller 2. For purposes of explanation, it is

assumed herein that the two lowest order address bits on the bus 57 have the following relationship to the I/O units: 00 is for device A, 01 is for device B, 10 is for device C and 11 is for device D. The I/O commands appearing on the bus 56 are stored at the locations in the register file 25 according to the device addresses and, hence, the I/O devices for which they are intended. Thus, any I/O command for device A is stored at the 00 locations, any command for device B is stored at the 01 location, etc.

The microprocessor 11 controls the readout of the I/O command from the register file 25. In particular, the microprocessor 11 executes a memory read or move wherein the address it places on the microprocessor address bus 38 is such as to produce a chip select 5 (CS5) signal and the two lowest order address bits on the microprocessor address bus 38 are of the appropriate value to select the desired byte in the register file 25. The CS5 signal is supplied to the read enable terminal of the register file 25 and the two lowest order address bits are supplied to the read select terminals of the register file 25. This causes a readout of the desired I/O command which is then transferred to the microprocessor 11 via a bus 62 and the microprocessor data bus 37.

The I/O controller 2 further includes address selector circuitry 63 for selectively transferring address bits from either the microprocessor address bus 38 or the host processor channel address bus 35 to the address circuitry of the random access storage mechanism 22a, 22b. Thus, the storage units 22a and 22b can be addressed by either the host processor 1 or one of the address producing units connected to the microprocessor bus 16. In the present embodiment, each of the microprocessor 11 and the DMA controller 13 can supply addresses to the microprocessor bus 16 for purposes of addressing the storage units 22a and 22b.

This microprocessor bus addressing is indicated by the buses 64 and 65 which, respectively, connect the microprocessor address buses 38a and 38b to one set of input terminals of the address selector 63. The other set of input terminals of the address selector 63 are connected to the internal address buses 56 and 57 which are, in turn, connected to the host processor address buses 35a and 35b, respectively. Under the control of a control signal from the storage control logic 48, by way of line 67, the address selector 63 selects which of these two address inputs is to be supplied to the address terminals of the storage units 22a, 22b. As indicated by the common address bus 66, the address value appearing at the output terminals of the address selector 63 is always supplied to both of the storage units 22a and 22b. Further details of the address selection process will be discussed hereinafter in connection with Figure 11.

For sake of example, it is assumed that the

address selector 63 is comprised of four 2-line-to-1-line, data selector units. Each such unit is capable of handling the 2-to-1 selection for four output lines. Thus, four such units can handle the 2-to-1 one selection for 16 output lines. As will be seen, some of these 16 possible output lines are not used for addressing the storage units 22a and 22b and, hence, are left disconnected.

For the present moment, it can be assumed that the allow host connect signal produced by the storage control logic 48 and supplied to the address selector 63 by way of line 67 is produced in response to the address gate capture signal and corresponds to such address gate capture signal. This is approximately correct and will suffice for purposes of the explanation being given at this point. The binary one level of this allow host connect signal on line 67 causes the address selector 63 to switch to the left hand input terminals to supply the host processor address bus bits to the storage units 22a and 22b. Conversely, when the allow host connect signal is not present on the line 67, this line goes to a binary zero value and the address selector 63 is switched to select the right hand input terminals to connect the microprocessor address bus bits to the storage units 22a and 22b.

As can be seen from the foregoing, the appearance of the address gate capture signal on line 24 causes three primary things to happen. First, it enables the device select decoder 60 to supply a unique device indicative signal to the programmable interrupt controller 14. Secondly, it causes the I/O command to be stored into the register file 25. Thirdly, it causes a host processor derived address value to be supplied to the storage units 22a and 22b. This, together with related control signal supplied to the drivers 44 and 45 and the data register 46, enables the data word portion of the IDCB to be transferred from the host processor channel data bus 34 to the storage units 22a and 22b, or vice versa.

Thus, all four bytes of the immediate device control block (IDCB) are digested by the I/O controller 2 during one and the same interval of time, this being the interval of time at which they are placed on the host processor channel bus 8 by the host processor 1. Also, this digesting of the IDCB bytes is transparent to the microprocessor 11. The microprocessor 11 does not know that it has happened, except at such later time as it receives and accepts an interrupt request from the programmable interrupt controller 14. The fact that the I/O controller 2 is always capable of accepting the immediate device control block when it is presented by the host processor means that the I/O controller 2 never has to return a "controller busy" signal to the host processor 1. This, of course, prevents lost time on the part of the host processor 1.

The I/O controller 2 also includes a chip

select decoder 68 which is responsive to the higher order address bits on the microprocessor address bus 38a for decoding same to produce various chip select signals CS0, CS1, CS2, ..., CSn. These chip select signals are used to select or enable different ones of the other units in the I/O controller 2. Thus, for example, the chip select signal CS0 is used to select the control program storage 12, the chip select signal CS1 is used by way of an AND circuit 69 to enable the two-way drivers 41 and the chip select signal CS2 is used by way of an AND circuit 70 to enable the two-way drivers 40. The CS1 and CS2 signals are also used to select between the two different ones of the storage units 22a and 22b, this being accomplished by way of the storage control logic 48. In this regard, the CS3 chip select signal is used via the logic 48 to simultaneously select both of the storage units 22a and 22b. This simultaneous selection is done for purposes of transferring data words between the host processor data bus 34 and the storage units 22a and 22b during the cycle stealing operations controlled by the DMA controller 13. Thus, it is the DMA controller 13 which supplies the address to the chip select decoder 68 to produce the CS3 chip select signal.

Referring to Figure 2D, there is shown examples of four specific I/O units that can be attached to the microprocessor bus 16. The first I/O unit is comprised of a keyboard and cathode ray tube (CRT) display device 71 and its associated serial communication interface 72. The second I/O unit is comprised of a keyboard and CRT display device 73 and its associated serial communication interface 74. The third I/O unit is comprised of a wire matrix printer 75 and its associated printer controller 76. The fourth I/O unit is comprised of a floppy disk storage unit 77 and its associated floppy disk controller 78. Each of the serial communication interface 72 and 74 can be a universal synchronous/asynchronous receiver/transmitter (USART). The printer controller 76 can be a microprocessor. The floppy disk controller 78 can be a programmable floppy disk controller.

The foregoing particular types of I/O devices and device control units are intended as examples only. A wide variety of different makes and types of I/O units are available and can be attached to the microprocessor bus 16 in place of one or more of those described above.

Operation of the Figure 2 embodiment

Considering briefly a typical data transfer operation for the I/O controller 2 of Figure 2, it is initially noted that successive data bytes received from an I/O device are alternately stored in the high byte storage unit 22a and the low byte storage unit 22b. Considering the case where data is being transferred from an I/O device to the host processor 1 and assuming, for example, the I/O device is the keyboard/dis-

play unit 71, the data bits are transmitted serially from the keyboard display unit 71 to the serial communication interface (serializer/deserializer) 72. After the first byte of data is assembled in the interface 72, it is transferred by way of the microprocessor data bus 37 to the microprocessor 11 and then from the microprocessor 11 via the data bus 37 and the two-way drivers 40 to the high byte storage unit 22a. After the second byte of data is assembled by the interface 72, it is transferred by way of the microprocessor data bus 37 to the microprocessor 11 and then by way of the microprocessor data bus 37 and the two-way drivers 41 to the low byte storage unit 22b. Subsequent successive bytes are alternately stored in this same manner in the high byte storage unit 22a and the low byte storage unit 22b, the third, fifth, seventh, etc., bytes being stored in the high byte storage 22a and the fourth, sixth, eighth, etc., bytes being stored in the low byte storage 22b.

After the desired amount of data for the I/O device in question has been accumulated in the dual port storage units 22a and 22b, such data is thereafter transferred to the host processor 1. When data is transferred to the host processor 1, it is transferred to such host processor two bytes or one word at a time. In other words, a high order byte is read out of the storage unit 22a simultaneously with the readout of a low order byte from the storage unit 22b and both such bytes are simultaneously transferred to the host processor 1 by way of the two-way drivers 45 and 44 and the host processor channel data bus 34. As an intermediate step, the two-byte data word may be temporarily stored in the 16-bit data register 46. This intermediate step (which won't be performed if the host processor responds fast enough) enables adjustment of the controller timing to the host interface handshake timing.

When data is being transferred in the opposite direction, namely, from the host processor to the I/O device, then the opposite kind of thing happens. Each 2-byte data word is sent to the I/O controller 2 via the host processor data bus 34 and the high order byte is stored in storage unit 22a and the low order byte is stored in storage unit 22b. The data bytes subsequently transferred to the I/O device are alternately taken from the high byte storage unit 22a and the low byte storage unit 22b.

The use of the separate high byte and low byte storage units 22a and 22b provides an automatic byte-to-word (or word-to-byte) formatting which is one of the novel inventive features which is incorporated in the I/O controller 2.

In the Figure 2 embodiment, the transfer of data between the I/O device and the dual port storage units 22a and 22b is controlled by the microprocessor 11. When the I/O device is ready for the transfer of a data byte, it supplies an interrupt request (IR) to the programmable

interrupt controller (PIC) 61. PIC 61 then sends an interrupt request to the microprocessor 11 which thereafter causes the microprocessor 11 to perform the necessary instruction routine for transferring a byte of data from the dual port storage 22 to the I/O device or vice versa. The interrupt line running to the microprocessor 11 is one of the control lines in the microprocessor control bus 39.

In the Figure 2 embodiment, data is normally transferred between the host processor 1 and the dual port storage units 22a and 22b in a cycle steal mode. These cycle stealing operations are controlled by the DMA controller 13 and host DMA request logic 28. The internal construction of the DMA controller 13 will be hereinafter described in connection with Figure 8. As will be seen, such DMA controller 13 includes four separate address counters which are normally used to perform DMA operations for four different I/O devices. One of the DMA address counters is used to keep track of the host processor main storage address and another of the DMA address counters is used to keep track of the addresses for the dual port storage units 22a and 22b. Whenever a particular DMA request (DRQ) line is activated, the DMA controller 13 puts the corresponding address counter address onto the 16-bit microprocessor address bus 38. If two or more DMA request (DRQ) lines are active at the same time, then a priority resolver inside the DMA controller 13 selects and processes the requests one at a time in the appropriate order.

Cycle steal transfer operations are controlled by use of the channel 0 and channel 1 circuit in the DMA controller 13. The channel 0 address counter 82 (Figure 8) supplies the host processor main storage addresses and the channel 1 address counter 84 is used to provide the addresses which are supplied to the dual port storage unit.

For any given data word transfer, the DMA request logic 28 first activates the DRQ 0 line. This causes the DMA controller 13 to put the host processor main storage address to be used for this data word transfer onto the microprocessor address bus 38. This address is then strobed into the cycle steal address register 27, whereafter it is placed on the 16-bit host processor address bus 35 by way of the two-way drivers 51. At the appropriate point after the main storage address is strobed into the cycle steal address register 27 the DRQ 1 request line is activated and/or recognized by the DMA controller 13 to cause the DMA controller 13 to put the storage address for the dual port storage units 22a and 22b onto the microprocessor address bus 38. This address is then transferred to the storage units 22a and 22b by way of the address selector 63. This address is the dual port storage address to or from which the data word is transferred.

The foregoing DRQ 0/DRQ 1 sequence is repeated for each data word transferred. The two

DMA address counters which are being used for these transfers are incremented after each data word transfer. The host DMA request logic 28 also initiates the generation of the cycle steal requests which are sent to the host processor 1, each such request being recommenced immediately after the leading edge of the DRQ 0 signal is sent to the DMA controller 13.

Figure 7 is a timing diagram showing what happens for a more or less typical instruction cycle for the microprocessor 11. The particular instruction cycle which is shown in Figure 7 is for the OUT instruction which causes the content of the microprocessor accumulator register to be placed on the microprocessor data bus 37 for transmission to the I/O port specified by the address placed on the microprocessor address bus 38. Actually, the term "I/O port" in the previous sentence is inaccurate in the sense that the content of the accumulator register will be transmitted to any element or device which is connected to the microprocessor bus and which is responsive to or selected by the particular "port" address appearing on the microprocessor address bus 38.

An interesting peculiarity of the microprocessor is that the low order address bits (bits 0—7) are time multiplexed on the data bus output. As indicated in Figure 2B these low order address bits are immediately latched into an 8-bit latch 80 by the address latch enable (ALE) output pulse of the microprocessor. Latch circuits 80 in turn drive the low order 8-bit microprocessor address bus 38b.

The IO/M decoder 81 is shown in Figure 2B is used to convert the microprocessor RD, WR and IO/M output signals into the following four more conventional signals: MEMR, MEMW, IOR and IOW, which respectively stand for memory read, memory write, I/O read and I/O write. These four signals produced by the IO/M decoder 81 are supplied to and appear on four separate and additional control lines of the microprocessor control bus 39. These signals are used by various other of the units in the I/O controller 2. As will be seen, the DMA controller 13 is also connected to these four additional control lines and can also produce these MEMR, MEMW, IOR and IOW signals.

The READY control terminal of the microprocessor 11 is of particular interest. It provides a means for enabling external circuitry to place the microprocessor 11 in a "wait" state. More particularly, the microprocessor 11 includes internal circuitry which is responsive to the absence of the external READY signal for placing the microprocessor 11 in a "wait" state. When the READY signal reappears, the microprocessor 11 resumes its operations just as if nothing had happened.

Description of the Figure 8 DMA controller

Figure 8 shows in greater detail one possible form of internal construction for the direct

memory access (DMA) controller 13 of Figure 2. The abbreviations used in Figure 8 are as follows:

Term	Meaning
IOR	I/O Read
IOW	I/O Write
CS	Chip Select
HRQ	Hold Request
HLDA	Hold Acknowledge
MEMR	Memory Read
MEMW	Memory Write
AEN	Address Enable
ADSTB	Address Strobe
TC	Terminal Count
DRQ	DMA Request
DACK	DMA Acknowledge

The DMA controller 13 includes four separate channels, namely, channels 0—3, which are normally used to perform DMA operations for four different I/O devices. Each channel includes its own address counter and byte counter. Thus, channel 0 includes address counter 82 and byte counter 83, channel 1 includes address counter 84 and byte counter 85, channel 2 includes address counter 86 and byte counter 87 and channel 3 includes address counter 88 and byte counter 89. The purpose of each address counter is to provide for a particular I/O device the addresses needed for addressing a storage unit, these addresses being the addresses of the storage locations which are to receive data from or supply data to the particular I/O device in question. The purpose of each byte counter is to provide for its particular I/O unit a count showing the number of bytes remaining to be transferred for the usual case where a multi-byte block of data is to be transferred. With respect to a particular I/O device, its DMA address counter is incremented and its byte counter is decremented after each byte is transferred for such I/O device.

The DMA controller 13 also includes a priority resolver 90 for handling the case where two or more of the DMA request (DRQ) input lines are active at the same time. In such case, the priority resolver 90 selects and processes the requests one at a time in the appropriate order. As will be discussed in greater detail hereinafter, the DMA controller 13 is operated in its rotating priority mode when used in the I/O controller embodiment of Figure 2. In this rotating priority mode, the priority of the channels has a circular sequence. After each DMA channel is serviced, the priority of each channel changes. The channel which has just been serviced will be given the lowest priority.

Figure 9 is a timing diagram showing the various DMA signal waveforms for the case of two typical successive DMA cycles. When not active, the DMA controller 13 sits in an idle state (SI). The DMA controller 13 time multi-

plexes some of the address bits out of its data bus output in a manner similar to that which was done by the microprocessor 11. In the case of the DMA controller 13, however, it is the higher order address bits (8—15) which are multiplexed. As indicated in Figure 2B, these high order address bits are immediately latched into a set of 8 latch circuits 91 by the address strobe (ADSTB) signal which is generated by the DMA controller 13.

A further point to note is that, when the DMA controller 13 is active, the operation of the microprocessor 11 is suspended by placing it in a "hold" state. More particularly, shortly after the DMA controller 13 receives a DMA request (DRQ), it sends a hold request (HRQ) to the HOLD terminal of the microprocessor 11 via the HOLD line of the microprocessor control bus 39. When the microprocessor 11 enters the "hold" state and so long as it remains in that state, it supplies a hold acknowledgment (HLDA) signal to the DMA controller 13. During this HLDA interval, the data and address outputs of the microprocessor 11 are placed in a three-state of high impedance condition so as not to affect the microprocessor data and address buses 37 and 38. Also, the output of the microprocessor address latch 80 and the IO/M decoder 81 are placed in the high impedance condition by the address enable (AEN) signal. Thus, during the HLDA interval, the DMA controller 13 can place addresses on the microprocessor address bus 38 and the devices or elements which respond to these addresses can place data on the microprocessor data bus 37 without interference from the microprocessor 11. The placement of data on and the reading of data from the microprocessor data bus 37 is controlled by the read and write pulses produced by the DMA controller 13.

The READY input control line for the DMA controller 13 provides the same function for the DMA controller 13 as was provided for the microprocessor 11 by its READY input control line. More particularly, when the READY signal is present (READY line at the binary one level), the DMA controller 13 operates in its normal manner. When, on the other hand, the READY signal is absent (READY line at a binary zero level), the DMA controller 13 will go into a "wait" state and will wait for the READY signal to reappear before completing the current DMA cycle. In the typical application, this "Not Ready" function is used to elongate the storage read and storage write cycles with Wait states for the case of relatively slow storage units.

Description of Figure 10 interrupt and cycle steal handshaking unit

Referring now to Figure 10A and B, there is shown in greater detail the internal construction of the interrupt and cycle steal handshaking unit 33 of Figure 2C. Figure 10B should be placed on the right side of Figure 10A to form a single figure which will be referred to

herein as Figure 10. This interrupt and cycle steal handshaking unit handles the handshaking signaling sequences with the host processor 1 for two different cases. The first case is where the I/O controller 2 presents a cycle steal request to the host processor 1 for purposes of cycle stealing a data word into or out of the host processor main storage unit 7. The other case is where the I/O controller wants to present an interrupt request to the host processor 1.

Considering first the case of a cycle steal request, a cycle steal request signal is received from the host DMA request logic 28 by way of line 31 and is used to set a cycle steal request latch 92. This produces a cycle steal request signal at the output of latch 92, which signal is transferred by way of an AND circuit 93 to a cycle steal request in line in the host processor control bus 36. In due course, the host processor 1 recognizes this cycle steal request by sending out a 5-bit poll ID on the poll ID bus portion 94 of the channel control bus 36 and a poll signal on the control bus poll line 95.

A unique poll ID is used for responding to cycle steal requests and this unique poll ID is decoded by a cycle steal poll decoder 96. Decoder 96 then produces an output signal which sets a cycle steal compare latch 97. This produces a one level output at the output of latch 97 which is supplied by way of an OR circuit 98 to a first input of an AND circuit 99. The second input of AND circuit 99 receives the poll signal via OR circuit 100. These two signals being present at the two inputs of the AND circuit 99 causes a poll return latch 101 to be set. This supplies by way of an AND circuit 102 a poll return signal to the host processor 1 which tells the host processor 1 that the I/O controller is ready to proceed.

The set condition of the poll return latch 101 plus the presence of the poll signal at the output of OR circuit 100 plus the set condition of the cycle steal compare latch 97 causes a cycle steal poll capture latch 103 to be set via AND circuit 103a. After receipt of the poll return signal and when it is ready to do the data word transfer, the host processor 1 sends out a service gate signal on the service gate control line 104 of the channel control bus 36. This service gate signal, together with a set condition of latch 103, causes a cycle steal service gate capture latch 105 to be set via AND circuit 106. This starts the cycle steal service gate capture signal on the latch output line 32, which signal is used to control the operation of various other units in the I/O controller 2. This service gate capture signal on line 32 is terminated by the trailing edge of the service gate signal via NOT circuit 107.

The output of the service gate capture latch 105 is also supplied to a first input of an AND circuit 108. The second input of AND circuit 108 is controlled by latch 109. Latch circuit 109 is set via AND circuit 110 by the DACK 0

and MEMW signals which occur when the main storage address is strobed into the cycle steal address register 27. In other words, they occur when the host processor main storage address is ready and available to be sent to the host processor 1. The set condition of latch 109, plus the presence of the cycle steal service gate capture signal at the first input of AND circuit 108, produces a service gate return signal which is supplied by way of OR circuit 111 to the service gate return line in the channel control bus 36. This service gate return signal tells the host processor 1 that everything is proceeding according to schedule.

Subsequent to receipt of this service gate return signal, the host processor 1 sends out a data strobe signal on the host data strobe line 112 in the channel control bus 36. This data strobe signal is transferred by way of a driver circuit 113 to a line 114 which makes it available to the other units in the I/O controller 2. This data strobe signal is used, for example, to control the strobing of the data into the 2-byte data register 46 shown in Figure 2A.

Considering now the interrupt portion of Figure 10, the microprocessor 11 initiates an interrupt request by doing an I/O port type OUT instruction with the appropriate address and data bus values to cause activation of an AND circuit 115, which in turn causes an interrupt request latch 116 to be set. In other words, the address produced by the microprocessor 11 produces a CS8 chip select signal on line 211 and the data value on the microprocessor data bus is such that data bit 2 has a one value on line 212. Before proceeding further, it is necessary to briefly consider the manner in which interrupt operations are handled. In particular, a priority type interrupt system is used wherein each of the various I/O units attached to it are assigned one of several possible priority levels. In this scheme of things, the I/O unit presenting the interrupt must present its interrupt on a certain priority level interrupt line. This is accomplished by loading in advance into the I/O unit the priority level value to be used by such I/O unit.

In Figure 10, this priority level value is loaded into the priority level register 117. This is accomplished by issuing a so-called "Prepare" command to the I/O controller 2, this command on line 214 being detected by a decoder 118 to activate an AND gate 119, which is at that time receiving an address gate capture signal at its other input by way of line 213, with the output of the AND gate 119 being used to strobe into the priority level register 117 the desired priority level value then appearing on the channel data bus 34. The priority level value in register 117 drives a decoder 120 having multiple output lines, only one of which is activated in accordance with the priority level value supplied by the register 117. The multiple output lines of the decoder 120 are supplied by way of AND gate 121 to the multiple interrupt

request in lines in the channel control bus 36. Only the particular interrupt request in line corresponding to the priority level value in the register 117 is activated by the decoder 120. The actual moment of presentation of the interrupt request to the host processor 1 is controlled by the interrupt request latch 116. When this latch 116 is in its set condition, it supplies a one level signal by way of AND circuit 122 to the AND gate 121. This supplies the interrupt request to the host processor 1.

In due course, the host processor 1 recognizes the interrupt request and responds thereto by sending out a poll ID on the channel control bus portion 94 and a poll signal on the channel control bus line 95. The value sent out on the poll ID bus 94 is the value of the interrupt level being used by this I/O controller. This value is compared in the interrupt poll compare unit 123 with the priority level value in the priority level register 117. If the priority level values match, then the interrupt poll compare circuit 123 produces a one level output which is supplied by way of AND circuit 124 to set an interrupt compare latch 125, provided a valid interrupt request is pending as indicated by the one level output of the interrupt request latch 116. Assuming the interrupt compare latch 125 is set, then this enables the poll return latch 101 to be set by the poll signal on channel control line 95. The setting of latch 101, as before, generates the poll return signal which is supplied back to the host processor 1.

The set condition of the interrupt compare latch 125 is supplied by way of an AND circuit 126 to set an interrupt poll capture latch 127, the other input to the AND circuit 126 at this time being at the binary one level.

In response to the poll return signal, the host processor 1 in due course sends out a service gate signal on the channel control line 104. This activates the second input to an AND circuit 128 which, together with the set condition of the interrupt latch 127, causes a setting of an interrupt service gate capture latch 129. This produces an interrupt service gate capture signal on line 130, this signal being used by other units in the I/O controller 2. The interrupt service gate capture latch 129 is reset by the trailing edge of the service gate signal on line 104 via NOT circuit 131.

The output of the interrupt service gate capture latch 129 is also supplied to an AND circuit 132, the other input of which is assumed to be activated at this moment by an allow host connect signal (to be discussed hereinafter) on line 215. The resulting one level at the output of AND circuit 132 is supplied by way of the OR circuit 111 to provide the service gate return signal on the service gate return line in the channel control bus 36. This service gate return signal tells the host processor 1 that everything is proceeding according to schedule.

A point to note from the foregoing is that two different types of service gate capture signals

are produced, one being produced for the cycle steal of a data word to or from the host processor 1 and the other being produced when the I/O controller 2 desires to interrupt the host processor 1.

Description of Figure 11 storage control and address selection logic

Referring now to Figures 11A and B, there is shown in greater detail the internal construction of the storage control logic 48 of Figure 2A, this being the control logic for the dual port storage 22a, 22b. Figure 11B should be placed on the right side of Figure 11A to form a single figure which will be referred to herein as Figure 11. A first part of the Figure 11 logic can be thought of as being the "address selection logic". This portion is represented by circuit elements 135—142. This 135—142 logic controls the address selector 63. When the output of flip-flop 138 is at the zero level, the address selector 63 connects the microprocessor address bus 38 to the address lines 66 running to the high byte and low byte storage units 22a and 22b. The same address, namely the address appearing at the output of address selector 63 is always supplied to each of these storage units 22a and 22b.

The purpose of the address selection logic 135—142 is to enable alternative addressing of the dual port storage 22a, 22b for host processor DPC (Direct Program Control) and other IDCB transfer operations and for controller initiated interrupt request operations.

A second portion of Figure 11 logic is represented by circuit elements 143—147. These elements control the "select" and "write" lines running to the storage units 22a and 22b. Its "select" line must be active in order for a storage unit to read in or write out any data.

At this point, it is helpful to consider the memory address range map shown in Figure 12. This is a map of the lower half (0—32K) of the total address range capable of being addressed by the 16-bit microprocessor address bus 38. As seen from Figure 12, the second 8K (2000—3FFF in hexadecimal) of the addressing range is used for the low byte storage 22b and the third 8K (4000—5FFF hexadecimal) is used for the high byte storage 22a. As indicated on the left side of Figure 12, the three highest order bits of the 16-bit microprocessor address bus 38 are used for chip select purposes. These three highest order microprocessor address bits are not supplied to the address selector 63. They are instead supplied to the chip select decoder 68 which is shown in Figure 2B and which generates chip select signals CS1, CS2 and CS3, which signals are supplied to the dual port storage unit select logic represented by circuit elements 143—145 in Figure 11. Thus, if the address on the microprocessor address bus 38 is in the 2000—3FFF range, the chip select signal CS1 on line 216 is active via OR gate 143 to select

the low byte storage 22b. If the micro-processor bus address is in the 4000—5FFF range, then chip select signal CS2 on line 217 is active via OR gate 144 to select the high byte storage 22a. If the MP bus address is in the 6000—7FFF range, then the chip select signal CS3 is active via OR gate 145 and both of OR gates 143 and 144 to select both the high byte storage 22a and the low byte storage 22b. This provides the read-in or write-out of a complete 2-byte data word. Note with respect to Figure 12 that MP bus addresses in the 6000—7FFF range are used only for chip selection purposes and that there is no separate physical storage provided for this part of the address range.

For cycle steal operations, the storage "write" line is controlled by the MEMW (memory write) signal via OR gate 146. This signal is obtained from the DMA controller 13 via the MP control bus 39. If the storage "write" line is not active, then the occurrence of a storage select signal will enable the storage unit to do a read-out operation. (If "write" is off, the contents of the addressed storage location are placed on the storage data bus by the occurrence of the select signal).

A further thing to note from Figure 12 is that the 32 lowest byte locations in each of storage 22a and 22b are reserved for and used only for host processor I/O command (IDCB) transfer operations. Figure 13 is a blow-up or enlargement of the storage maps for these 32 lowest byte locations. The map of Figure 13 applies to each of the byte wide storage units 22a and 22b.

The significance of what is meant by DPC (Direct Program Control) operations is indicated in Figures 3 and 6. Each EPC read command enables a 2-byte word of data or status information to be transferred from the I/O controller 2 to the host processor 1. Each DPC write operation enables a word of data or control information to be transferred from the host processor 1 to the I/O controller 2. The DPC data word (IDCB bits 16—31) is transferred by way of the channel data bus 34. This DPC data word is stored into (or transferred out of) the dual port storage 22a, 22b, with the higher order byte (Byte 2 or bits 16—23) being stored in or read from the high byte storage 22a and the lower order byte (Byte 3 or bits 24—31) being stored in or read from the low byte storage 22b. This type of data transfer is called "DPC" because the transfer of each data word is under the direct control of the host processor program and the host processor must issue a separate I/O command for each word transferred.

When a host processor I/O command is received by and accepted by the I/O controller 2, the output of the flip-flop 138 of Figure 11 is turned on to activate the "select host" output of the AND gate 141. This switches the address selector 63 so as to connect the indicated host address bus bits to the 5 lowest order output

lines of the address selector 63, the remainder of the address selector inputs being grounded as indicated at 148. This causes the data word accompanying the I/O command to be stored into the dual port storage 22a, 22b in the manner indicated in Figure 13. Thus, if the I/O command is a type 1 write command for device A, the two bytes of the accompanying data word will be stored at the address=5 storage locations in storage units 22a and 22b. Host address bits 14 and 15 define which device it is and, as indicated in Figure 6, host address bits 1—3 define the operation type.

With reference to Figure 2A, the acceptance of a host processor I/O command by the I/O controller 2 is signified by the occurrence of an address gate capture signal at the output of the AND gate 54 associated with the address compare circuit 50. With reference to Figure 11, this address gate capture signal is supplied via line 24 and OR gate 135 to the AND gate 137. This enables the next occurring ALE pulse from microprocessor 11 or the next occurring address strobe (ADSTB) pulse from the DMA controller 13, whichever is the first to occur, to set the flip-flop 138. This produces the "Allow Host Connect" signal on line 150 which is connected to the output of flip-flop 138. This signal, via AND gate 141, sets the address selector 63 to the "Select host" position. The "Allow Host connect" signal is also supplied to OR gate 145 to cause, via OR gates 143 and 144, the "select" activation of both storage units 22a and 22b. The "Allow Host Connect" signal is also supplied to AND gate 147 to activate, via OR gate 146, the storage "write" line, provided the I/O command is a "write" type command (host address bit 1=1).

The use of the ALE and ABSTB pulses at OR gate 136 enables the host processor 1, in effect, to cycle steal the I/O command (IDCB) data word into the dual port storage 22a, 22b. In this regard, the dual port storage 22a, 22b is, in actuality, the "main" storage unit for the microprocessor 11 (and the DMA controller 13) and this mechanism enables the I/O command data word to be cycle stolen into such "main" storage without interrupting the program which is being executed by the microprocessor 11 (or DMA controller 13). This happens because the output of the flip-flop circuit 138 is also connected by way of a timer 151 and a NOT circuit 152 to the "Ready" inputs of both the microprocessor 11 and the DMA controller 13 via line 153 of the microprocessor control bus 39. In particular, when the "Allow Host Connect" signal goes to one, the output of NOT circuit 152 goes to zero, thus removing the Ready signal from the microprocessor and DMA controller. This causes each of the microprocessor 11 and DMA controller 13 to go into a "Wait" state. (Actually, either one or the other but not both of the microprocessor and DMA controller will be active at any given moment and the

Ready signal will affect only the active one of these two units).

This "not ready" condition will prevail for the length of time the flip-flop 138 is in the "set" state plus an additional length of time determined by the timer 151, this timer being in the nature of a one shot multivibrator. The flip-flop 138 is reset by the trailing edge of the address gate capture signal via OR gate 139 and NOT circuit 140. The additional time added by the timer 151 is dependent on the particular type of circuit technology that is used for the address selector 63 and the storage units 22a and 22b and in a typical application is selected to be equal to the time duration of approximately two microprocessor clock cycles. This additional time interval is added in order to enable the address selector 63 and its output lines 66 to settle down after the address selector 63 is switched back to the microprocessor address bus 38. When the "Ready" signal reappears at the output of the NOT circuit 152, then the previously operating one of the microprocessor and DMA controller resumes operation from the point at which its operation was suspended.

There is a third kind of sharing or multiplexing of the dual port storage addressing which now needs to be considered. This has to do with I/O controller to host processor interrupt request operations. After the interrupt request from the I/O controller 2 has been recognized by the host processor 1 and after the host processor 1 has established a connection with the I/O controller 2, the host processor 1 sends a service gate signal to the I/O controller 2. During this service gate interval, the host processor 1 takes in the data word appearing on the channel data bus 34 and this data word should be the interrupt ID word shown in Figure 5. This particular kind of service gate signal is called an interrupt service gate signal because it is sent out in response to an interrupt request (as opposed to a cycle steal request).

Receipt of the interrupt service gate signal by the I/O controller 2 is indicated by the occurrence of the interrupt service gate capture signal on line 130 which comes from the handshaking logic 33 of Figure 2C. This interrupt service gate (SG) capture signal also enables a setting of the flip-flop circuit 138 by the next occurring one of the ALE and ADSTB pulses. In this case, however, the output of flip-flop circuit 138 operates by way of AND gate 142 to activate the "high impedance output" control line of the address selector 63. This causes the address selector 63 to set each of its outputs to a three state or high impedance condition. This enables the +V voltage source to place all of the address lines 66 running to the storage units 22a and 22b at the binary one level. In other words, this effectively switches the storage address to a value of "1111....11". This addresses the top byte location in each of the storage units 22a and 22b. Referring to

Figure 12, it is seen that the top byte locations contain the data that is needed for the interrupt ID word of Figure 5. Thus, the proper ID word appears at the output of storage units 22a and 22b for transmission to the host processor 1 during the interrupt service gate interval.

This data (the IIB byte and the device address) were previously loaded into these storage locations by the microprocessor 11 prior to issuance of the interrupt request to the host processor 1.

From the foregoing, it is seen that there are three different ways of addressing the dual port storage 22a, 22b. Furthermore, these three different ways are automatically multiplexed so as not to interfere with one another.

Figure 11 also shows the logic for operating the two-way drivers 44 and 45. This logic is represented by circuit elements 154—160. Each of two-way drivers 44 and 45 has two control inputs, namely, an output enable (OE) control terminal and a direction (D) control terminal. Activation of the OE input enables the output of the driver so that whatever signal is being supplied to the driver input appears at its output. When OE is inactive or off, the driver outputs are set to a three state or high impedance condition. When the direction control line D is active, it reverses the normal direction for movement of data through the driver. In the present embodiment, the normal direction for drivers 44 and 45 is from left to right. When the direction input D is active, data can pass in the reverse direction, namely, from right to left.

For the two-way driver 45, the output enable line OE3 is activated whenever both of storage units 22a and 22b are simultaneously "selected". This is accomplished by the output of OR gate 145. The direction line D3 is activated to allow data movement from right to left when AND gate 160 supplies a "storage (22a, 22b) to register (data register 46)" signal or when AND gate 157 provides an IDCB read signal, the former being for cycle steal operations and the latter being for I/O command transfer operations.

The output enable line OE1 for the two-way drivers 44 is activated during the occurrence of a cycle steal service gate capture signal on line 32 or the occurrence of an allow host connect signal at the output of flip-flop 138. The direction control line D1 is activated to enable movement of data from right to left through the driver 44 when either a "register to host" signal is generated by AND circuit 155 or an IDCB read signal is generated by AND gate 157.

Description of Figure 14 host DMA request logic and data register control

Referring now to Figure 14, there is shown the details for both the host DMA request logic 28 of Figure 2B and the data register control logic 47 of Figure 2A. The data register control logic 47 appears in the lower portion of Figure

14 and is enclosed by the dash lined box. The upper portion of Figure 14 is the host DMA request logic 28.

The host DMA request logic 28 controls the DMA controller 13 which in turn controls the transfer of data between the host processor 1 and the dual port storage 22a, 22b in the cycle steal mode. With reference to the DMA controller details shown in Figure 8, these cycle steal transfer operations are controlled by use of the channel 0 and channel 1 circuit in the DMA controller 13. For the present embodiment, the channel 2 and channel 3 circuits are not used. The channel 0 address counter 82 supplies the host processor main storage addresses which are sent out over the host channel address bus 35 and the channel 1 address counter 84 is used to provide the addresses which are supplied to the dual port storage unit for moving the data from the dual port storage to the host processor data bus 34, or vice versa. The associated DMA byte counters 83 and 85 are used to keep track of the number of words remaining to be transferred. When the count in either counter goes to zero, it terminates the operation of the corresponding DMA channel.

Cycle steal operations are initiated by the host processor 1 sending out a start cycle steal command. In response thereto, the microprocessor 11 fetches from the host processor 1 the 8-word device control block (DCB). These DCB words are fetched in a cycle steal mode and are stored into the appropriate device section of the dual port storage 22a, 22b. Following completion of the DCB transfer, the microprocessor 11 uses the DCB information to initialize the DMA controller 13 for the main data transfer operation. In particular, it loads the DMA address counter 82 with the main storage starting address contained in the DCB word. The desired starting address for the dual port storage 22a, 22b is loaded into the second DMA address counter 84. This parameter is obtained from one of the instructions in the subroutine contained in control program storage 12 for setting up the DMA controller for cycle steal purposes for the particular device in question. The first two DMA byte counters 83 and 85 are loaded with the same value, namely, a value equal to one half of the byte count value contained in word 6 of the DCB. A factor of one-half is used because the byte counters are decremented by a value of one after each data word transfer, whereas the data word transfer constitutes the transfer of two bytes. If the DCB is set up to contain the word count instead of the byte count, then the factor of one-half need not be applied to the value loaded into the DMA byte counters.

A similar word versus byte factor enters into the operation of the DMA address counter 82. The addresses contained in this counter are main storage byte addresses and they are incremented by a value of one after each data

transfer. At the same time, the address loaded into the cycle steal address register 27 for each new cycle steal transfer should be two counts higher than the address previously loaded into such register 27 because each cycle steal transfer transfers a two-byte data word. This difference is taken into account by skewing the input lines to the cycle steal address register 27 one bit position to the left so as to effectively produce a left shift of one for the address bits as they are loaded into this register 27.

A pair of control bits contained in each of the DMA byte counters 83 and 85 are also initially loaded so as to tell the DMA controller whether it is to perform a read or a write operation. If it is a read operation (controller to host transfer), the control bits in the channel one counter 85 are set so that the DMA controller will produce a MEMR (memory read) pulse followed by an IOW (I/O write) pulse during the channel one DMA cycle. Conversely, if it is a write operation (host to controller transfer), then the control bits are loaded so that an IOR pulse followed by a MEMW pulse are generated during the channel one DMA cycle.

The appropriate control bit in the DMA mode register is initially loaded so that the priority resolver 90 will provide a rotating priority mode of operation. In this rotating priority mode, the priority of the different DRQ input lines has a circular sequence. After each DMA cycle, the priority of each DRQ line changes. The DRQ line which has just been serviced will be set to the lowest priority. As a consequence, if, for example, both DRQ 0 and DRQ 1 are turned on and left on, then the rotating priority mechanism will cause DRQ 0 and DRQ 1 to be serviced in an alternate manner, first one, then the other, then the first, etc.

After the initial set-up of the DMA controller 13, the commencement of the primary cycle steal data transfer operation is initiated by the microprocessor 11 and the host DMA request logic 28. In particular, the microprocessor 11 executes an I/O port OUT instruction which gives the data bits on the microprocessor data bus 37 special values and which places an address on the microprocessor address bus 38 such that a chip select CS7 signal is produced by the chip select decoder 68.

With reference to Figure 14, the data bit 7 on line 218 is given the desired value needed to set an I/O port latch 162 to provide the proper input/output indicator value. This data bit value is strobed into latch 162 by the chip select CS7 signal. Latch 162 is set to a one value when input (controller to host) cycle steal operations are to be performed. If, on the other hand, output (host to controller) cycle steal operations are to be performed, then latch 162 is loaded with a zero value.

This same microprocessor I/O port OUT instruction also puts the data bit 6 line 219 at a value of one. This bit together with the CS7 signal is applied to the AND gate 163 to cause

a flip-flop 164 to be placed in the "set" condition. This causes the commencement of the cycle steal operations. More particularly, the setting of flip-flop 164 turns on the start request line which, in turn, via OR circuit 165 turns on the DRQ 0 request line of the DMA controller 13.

Considering first the cause of output or host processor to I/O controller cycle steal operations (IN Latch 162 equal zero), the turning on of the DRQ 0 request line causes the DMA controller 13 to put the host processor main storage address on the microprocessor address bus 38. This address is strobed into the cycle steal address register 27 (Figure 2A) by the DACK 0 and MEMW pulses produced by the DMA controller 13 for the DRQ 0 DMA cycle. The one level of the DRQ 0 line also operates via AND gate 166 and OR gate 167 to supply the cycle steal request signal on line 31 which sets the cycle steal request latch 92 (Figure 10A) which, in turn, sends the cycle steal request in signal to the host processor 1. At this point it should be noted that the start request flip-flop 164 is reset via AND gate 168 by the DACK 0 and ADSTB pulses produced during the DRQ 0 cycle. This occurs late enough in the cycle so that the consequent turning off of the DRQ 0 line has no effect on the completion of the DRQ 0 cycle.

The DMA controller 13 and request logic 28 now sit and wait for the host processor 1 to recognize the cycle steal request and to send out its cycle steal service gate signal, the data word to be transferred also being placed on the host channel data bus 34 by the host processor 1 during this service gate interval. This occurrence is recognized by the DMA request logic 28 by means of the AND gate 169 located in the data register control logic 47. In particular, the occurrence of both the cycle steal service gate (CS/SG) capture signal on line 32 and the host data strobe signal on line 114 causes the AND gate 169 to produce a one-level output signal which is called a "host to data register" transfer signal. This signal is supplied by way of OR circuit 170 and the STB 2 line to strobe the host data bus data into the data register 46. The host to data register transfer signal at the output of AND gate 169 is also supplied by way of OR circuit 171 to set a flip-flop circuit 172. This turns on the DRQ 1 output line of flip-flop 172 to commence the DRQ 1 request to the DMA controller 13.

If the DMA controller 13 has finished the DRQ 0 cycle, then the DRQ 1 request is recognized and the DRQ 1 cycle commenced. Otherwise, the DRQ 1 request is held in abeyance until completion of the DRQ 0 cycle, at which point the DRQ 1 cycle is commenced.

During the DRQ 1 cycle, the DMA controller 13 puts the address for dual port storage 22a, 22b on the microprocessor address bus 38, the 13 lower order bits of this address being passed by the address selector 63 to the dual port

storage 22a, 22b. Shortly thereafter, the DACK 1 and MEMW signals from the DMA controller 13 are effective to produce a binary one level at the output of AND gate 173. This binary one level is called "data register to dual port store" transfer signal and is supplied by way of OR circuit 174 to the OE 2 input of the data register 46 to "enable" the output of the data register 46. This supplies the data word in the register 46 to the two-way driver 45 (Figure 11A) and hence to the data bus inputs of the high byte and low byte storage units 22a and 22b. The storage address put out by the DMA controller 13 during this DRQ 1 cycle is of such a value as to cause the production of a CS3 chip select signal by the chip select decoder 68. As indicated in Figure 11B this causes the "select" lines for both of storage 22a and 22b to be activated. Also, the occurrence of the MEMW signal during this DRQ 1 cycle activates the "write" control inputs of both storage units 22a and 22b. Thus, the data word supplied by the data register 46 and passed by the two-way drivers 45 is written into the high byte and low byte storage units 22a and 22b.

The occurrence of the DACK 1 signal during the DRQ 1 cycle is also supplied by way of AND gate 175 to set a flip-flop 176. The setting of flip-flop 176 turns the DRQ 0 request line back on again. Thus, after completion of the DRQ 1 cycle, another DRQ 0 cycle is commenced and the foregoing DRQ 0/DRQ 1 operations are repeated to cause a cycle steal transfer of the next data word. As mentioned, the DMA address counters 82 and 84 are incremented by one and the DMA byte counters 83 and 85 are decremented by one near the ends of their respective DRQ 0 and DRQ 1 cycles. Thus, new host processor and dual port storage addresses are provided for each new DRQ 0/DRQ 1 cycle repetition. These DRQ 0 and DRQ 1 cycles continue to be alternately repeated until the counts in byte counters 83 and 85 go to zero. When the count in counter 83 goes to zero, the DMA controller automatically shuts off the channel 0 operations and no more DRQ 0 cycles are performed. Similarly, when the channel one counter 85 goes to zero, channel one operations are terminated and no more DRQ 1 cycles are produced.

The cycle stealing of data in the opposite direction, namely, from the I/O controller 2 to the host processor 1 will now be considered. In this case, the IN latch 162 is set to one. After the set-up of the DMA controller 13, the cycle steal "IN" operations are commenced by the setting of flip-flop 164. This turns on DRQ 0. The binary one level of the DRQ 0 line immediately operates by way of AND gate 177 and OR circuit 171 to set flip-flop 172.

This turns on the DRQ 1 request line. The binary one level of the DRQ 1 line is immediately effective by way of AND gate 178 and OR circuit 167 to turn on the cycle steal

request line 31 to initiate the cycle steal request to the host processor 1.

Since both the DRQ 0 and DRQ 1 request lines are active, the DMA controller 13 will first perform a DRQ 0 cycle and will immediately follow it with a DRQ 1 cycle. The DRQ 0 cycle is effective to transfer the host processor main storage address from the DMA address counter 82 to the cycle steal address register 27. The DRQ 1 is effective to transfer the next data word from the high byte and low byte storage units 22a and 22b via the two-way driver 45 to the data register 46. In particular, the occurrence of the DACK 1 and MEMR signals during the DRQ 1 cycle causes an AND gate 179 to produce a one level output. This one level output is called a "dual port store to data register" transfer signal is supplied by way of OR circuit 170 to the data strobe input STB 2 of the data register 46. This strobes the data word into the data register 46.

The DMA controller 13 and the DMA request logic 28 now sit and wait for the host processor 1 to respond to the cycle steal request. (Actually, the host processor response may already be present at this point in time, in which case no waiting is required). The response and ready for data transfer condition of the host processor is indicated by the occurrence of the cycle steal service gate (CS/SG) capture signal on line 32. This capture signal produces a one-level output at the output of AND gate 180. This one-level output is called a "data register to host processor" transfer signal and is supplied by way of OR circuit 174 to the output enable line OE 2 of the data register 46. This, together with the enabled and right to left transfer condition of the two-way driver 44, places the data word on the host processor data bus 34 for transmission to the host processor 1.

The occurrence of the cycle steal service gate capture signal on line 32, together with the host data strobe from the host processor on line 114, causes an AND gate 181 to produce a one level output which in turn causes AND gate 182 to produce a one level output which in turn sets a flip-flop circuit 183. This turns the DRQ 0 request line back on again, such line having been turned off by the resetting of flip-flop 164 by AND gate 168 during the first DRQ 0 cycle. This turning back on of the DRQ 0 line causes a repeating of the above described DRQ 0 and DRQ 1 cycles and, hence, a transfer of the next data word to the host processor 1. These DRQ 0/DRQ 1 cycles continue to be repeated until the counts in DMA byte counters 83 and 85 go to zero, after which DMA operations are terminated and remain terminated until the issuance of a new start cycle steal command and a new device control block (DCB) by the host processor 1.

In the embodiment described up to this point, the DMA controller 13 was only involved in the cycle steal transfer of data words between the host processor 1 and the dual port

storage 22a, 22b. In such embodiment, the transfer of data bytes between the I/O device and the dual port storage 22a, 22b is handled by the microprocessor 11. A typical mode of operation would be for the microprocessor 11 to transfer a block or page of data, one byte at a time, from a given I/O device into the corresponding device section of the dual port storage 22a, 22b. After completion of this task, the microprocessor 11 would cause the issuance of an "attention" interrupt request to the host processor 1 to tell it that there was a block of data ready to be transferred to it. Thereafter, the host processor 1 would issue a start cycle steal command to the I/O controller 2. After performing the DCB fetch and the DMA controller set up, the DMA controller 13 would cause the cycle stealing transfer of the block or page of data from the dual port storage 22a, 22b to the host processor 1.

For the case of a typical data transfer in the opposite direction, namely, from the host processor 1 to an I/O device, a similar type of procedure would be applied in reverse, in this case the cycle stealing transfers from the host processor 1 to the dual port storage 22a, 22b occurring before the byte by byte transfer of the data from the dual port storage 22a, 22b to the I/O device.

Description of Figure 15 device DMA embodiment

Referring now to Figure 15, there is shown a modified embodiment whereby the data transfers between the I/O device and the dual port storage are also handled by the DMA controller 13. This embodiment can be used to overlap some of the device to dual port transfers with some of the dual port to host processor transfers or, conversely, to overlap some of the host processor to dual port transfers with some of the dual port to I/O device transfers.

In Figure 15, the device to dual port transfers (or vice versa) are handled by the DRQ 2 and DRQ 3 channels of the DMA controller 13. The DRQ 2 channel will be used to handle the transfers to or from the high byte storage 22a and the DRQ 3 channel will be used to handle the transfers to or from the low byte storage 22b. The purpose of the device DMA request logic 185 is to cause successive device DMA requests to be alternately applied to the DRQ 2 and DRQ 3 channels. This is necessary in order to cause successive data bytes to be alternately stored in the high byte storage 22a and the low byte storage 22b.

With reference to the DMA controller details in Figure 8, the channel 2 address counter 86 is initially loaded with the desired starting address in the high byte storage 22a, this address being in the appropriate address range to produce a CS2 chip select signal. The channel 3 address counter 88, on the other hand, is initially loaded with the appropriate starting address in the low

byte storage 22b, this address being in the address range needed to produce a CS1 chip select signal. In order to subsequently get the proper word type transfers, the starting addresses loaded into each of the counters 86 and 88 must be the same, except for the three highest order address bits which are used to do the chip selecting. The channel 2 and channel 3 byte counters are initially loaded with the appropriate values corresponding to the number of bytes to be transferred to the corresponding ones of the high byte storage 22a and the low byte storage 22b. As before, the DMA mode register is set so that the priority resolver 90 will operate in the rotating priority mode. As a result, after each DMA cycle, the priority of each channel will be changed with the channel which has just been serviced being set to the lowest priority.

With reference to Figure 15, when the peripheral device or I/O device 186 is ready for a data byte transfer, the device control unit 187 will raise its DMA request output line 220. This DMA request line runs to the device DMA request logic 185. The device request logic 185 is initially activated by means of an I/O port latch 188 which is loaded with a binary one value by the microprocessor 11 if it is desired for the data transfers for the peripheral device 186 to be performed in the DMA mode. Alternatively, latch 188 can be loaded with a zero value and the data transfers performed in the manner previously described. The loading of the latch 188 with a one value causes a flip-flop circuit 189 to be reset. This turns on the Q output which runs to the AND gate 190 and turns off the Q output which runs to the AND gate 191. This completes the initial set-up of the device request logic 185.

After initialization of the request logic 185, the first DMA request to be issued by the device control unit 187 will be passed by the AND gate 190 to the DRQ 2 input of the DMA controller 13. When the DRQ 2 channel gets its turn, this will cause a byte of data to be transferred from the device control 187 to the high byte storage 22a (or vice versa) via the microprocessor data bus 37. More particularly, the DACK 2 signal produced during the DRQ 2 cycle is supplied by way of OR circuits 192 and 193 to the chip select input of the device control 187 to cause or enable such device control 187 to put its data onto the microprocessor data bus 37. At the same time, the DACK 2 signal appearing at the output of OR circuit 192 is used to toggle the flip-flop 189 to its opposite state, in this case, its "set" state. This turns on the Q output and turns off the Q output. As a consequence, the next DMA request issued by the device control 187 is supplied by way of AND circuit 191 to the DRQ 3 input of the DMA controller 13. When this DRQ 3 request gets its turn, the DMA controller 13 will cause the next byte of data to be transferred from the device control 187 to the low byte storage 22b (or vice versa)

via the microprocessor data bus 37. The DACK 3 signal produced during the DRQ 3 cycle flips the flip-flop 189 to the opposite state and also enables the chip select input line of the device control 187.

Because of the flipping back and forth of flip-flop 189, successive DMA requests and data byte transfers are alternately handled by the DRQ 2 and DRQ 3 channels of the DMA controller 13.

A typical manner of operation for the Figure 15 embodiment will now be considered. For sake of example, this will be for the case where data is to be transferred from the I/O device to the host processor. In this example, the microprocessor 11 will initially activate the device request logic 185 but not the host request logic 28. This will cause a DMA transfer of a block or page of data from the I/O device 186 to the dual port storage 22a, 22b. After completion of this block transfer, the microprocessor 11 will activate both the host request logic 28 and the device request logic 185. This will cause a cycle stealing of the first block of data from the dual port storage 22a, 22b to the host processor 1 at the same time as a second block of data is being transferred from the I/O device 186 to the dual port storage 22a, 22b. Because of the rotating priority mode used by the DMA controller 13, the cycle steal transfers to the host processor will, in general, be interleaved with the DMA transfers from the I/O device to the dual port storage.

An advantage of this system is that one set of transfers need not wait on the other set. If, for example, the host processor 1 is busy with other tasks, then the DRQ 0 and DRQ 1 lines will be inactive. Nevertheless, the DRQ 2 and DRQ 3 lines can continue to transfer data from the I/O device 186 to the dual port storage 22a, 22b. Conversely, if for any given interval the I/O device 186 is not ready, then the DRQ 0 and DRQ 1 channels can nevertheless continue to cycle steal data to the host processor 1.

It is not necessary that the DRQ 0/DRQ 1 data transfers be for the same I/O device as are the DRQ 2/DRQ 3 data transfers. If desired, these two sets of transfers can be for two different I/O devices during the same time interval.

Description of Figure 16 multiple device DMA embodiment

Referring to Figure 16, there is shown an embodiment wherein the data transfers to the dual port storage 22a, 22b for three different I/O units are handled by DMA controllers. The I/O device 186 is handled by the DMA controller 13 in the manner previously described in Figure 15. In addition, two additional I/O devices 195 and 196 and their respective device controls 197 and 198 are handled by a second DMA controller 200. This second DMA controller 200 is of the same construction as the first DMA controller 13 and, as such, is also a 4-channel DMA

controller. A second device DMA request logic unit 201 handles the DMA request for the second I/O device 195 and a third device DMA request logic unit 202 handles the DMA requests for the third I/O device 196. Each of these device DMA request logic units 201 and 202 is of the same construction as the device DMA request logic 185 and both operate in the same manner as described above for the device DMA request logic 185.

The operation of the Figure 16 embodiment is relatively straightforward in view of the previous explanations except for the contention situation which arises when both the DMA controller 13 and the DMA controller 200 present "Hold" request signals (HRQ 1 and HRQ 2, respectively) to the microprocessor 11 at about the same time. This contention situation is resolved by the use of DMA contention logic 203. One possible form of construction for this contention logic 203 is shown in Figure 17.

Referring to Figure 17, the HRQ 1 and HRQ 2 signals from the controllers 13 and 200, respectively, are supplied to the correspondingly designated input terminals in Figure 17. At this point it is noted that, as shown in Figure 16, these two signals are also connected to the HOLD request line which is part of the microprocessor control bus 39 and which runs to the microprocessor 11, this connection to the HOLD request line being by way of an OR circuit 204. As indicated in Figure 17, the contention logic 203 receives back from the microprocessor 11 the Hold Acknowledge signal HLDA. The contention logic 203 then decides which of the two DMA controllers 13 and 200 is to get this HLDA signal. The one that gets it is allowed to proceed with its DMA operations, while the one that does not get it must sit and wait for its turn to come up.

The contention logic 203 includes AND circuits 205—207 and NOT circuits 208—210. The logic performed by these circuits is such that if HRQ 1 is at the 1 level and HRQ 2 is at the 0 level when the HLDA signal appears, then the HLDA signal goes to the first DMA controller 13 by way of the HLDA 1 output terminal. Conversely, if HRQ 2 is at the 1 level and HRQ 1 is at the 0 level when the HLDA signal appears, then such signal is sent to the second DMA controller 200 by way of the HLDA 2 output terminal. If, on the other hand, both HRQ 1 and HRQ 2 are at the 1 level when the HLDA signal appears, then this signal is sent to the first DMA controller 13 by way of the HLDA 1 output terminal.

Claims

1. A data interface mechanism for interfacing a M-byte data bus (8, 34) connected to a host processor (1) with an N-byte data bus (16, 37), where the ratio of M to N is an integer greater than 1, which N-byte data bus is con-

nected to I/O devices (3—6) by the intermediary of a microprocessor (11), comprising:

a number of random access storage units (22), the number of such storage units being equal to the ratio of M to N and each such storage unit having a width of N bytes,
common addressing circuitry (63) connected to the microprocessor's address bus (38) to receive first address bits and to supply said first address bits to all of the random access storage units,
logic circuitry (48) which is connected to receive a set of first control signals (CS1, CS2) and a further control signal (CS3) for activating a different one of said random access storage units in response to each of said first control signals, and for activating simultaneously all of said random access storage units in response to said further control signal, thereby enabling a storage operation to be carried out in the activated random access storage unit or units at the address specified by said first address bits;
a first data transfer mechanism (40, 41) which is connected to receive said set of first control signals and which, in response to each of said first control signals, couples the random access storage unit currently activated by the logic circuitry to the N-byte data bus for transferring an N-byte data segment between the N-byte data bus and said activated random access storage unit;
a second data transfer mechanism (44, 45) which is connected to receive said further control signal and which, in response thereto, couples simultaneously all of the random access storage units to the M-byte data bus for transferring in parallel an M-byte data segment between said M-byte data bus and said random access storage units; and
control signal generating means for generating said set of first control signals and said further control signal, said first control signals being generated in an order such that said random access storage units are accessed one at a time in a rotating manner by said logic circuitry;
said data interface mechanism being characterized in that:
said first address bits form the lower order address bits of an address on the microprocessor's address bus (38);
said control signal generating means comprises decoder circuitry (68) which is connected to receive the higher order address bits of an address on the microprocessor's address bus and which, in response to the higher order address bits of an address in one of a set of first address ranges, generates a respective one of said first control signals and, in response to the higher order address bits, of an address in a further address range, generates said further control signal; and
said microprocessor is provided with means for

placing on said microprocessor's address bus a series of first addresses, each of which is an address in one of said set of first address ranges, to cause the transfer of data in N-byte segments between the random access storage elements and the N-byte data bus, and for placing on said microprocessor's address bus an address in said further address range to cause the transfer in parallel of M bytes of data between the random access storage units and the M-byte data bus.

2. Data interface mechanism according to claim 1 in which $M=2$ and $N=1$, two random access storage units being located intermediate the two-byte bus (34) and the one-byte bus (37), with each such storage unit having a width of one byte.

3. Data interface mechanism in accordance with claim 1 or claim 2 characterized in that it further includes a direct memory access controller (13) for supplying some of the addresses used for accessing said storage units (22).

Patentansprüche

1. Daten-Schnittstellen-Mechanismus zum Verbinden eines mit einem Hauptspeicher (1) gekoppelten M-Byte-Datensammelbus (8, 34) mit einem N-Byte-Datensammelbus (16, 37), wo das Verhaeltnis M:N eine hoehere als 1 Ganzzahl ist, wobei der N-Byte-Sammelbus ueber eine Mikroprozessor (1) an E/A-Einheiten (3—6) gekoppelt ist, das aufweist:

- Eine Anzahl von Speichern mit wahlfreiem Zugriff (22), wobei die Anzahl solcher Speichereinheiten dem Verhaeltnis M:N gleich ist, und jede Speichereinheit eine Breite von N-Bytes aufweist,
- eine gemeinsame Adressierungsverbindung (63), die an dem Adressen-Sammelbus (38) des Mikroprozessors gekoppelt ist, um erste Adressbits zu empfangen und die ersten Adressbits zu allen Speichereinheiten mit wahlfreiem Zugriff zu uebertragen,
- einen logischen Schaltkreis (48), der zum Empfang eines Satzes von ersten Steuersignalen (CS1, CS2) und eines weiteren Steuersignals (CS3) zur Aktivierung einer anderen Speichereinheit mit wahlfreiem Zugriff als die obengenannten Speichereinheiten im Ansprechen auf jedes der ersten Steuersignale und zur gleichzeitigen Aktivierung aller Speichereinheiten mit wahlfreiem Zugriff im Ansprechen des weiteren Steuersignals verbunden ist, wodurch Durchfuehrung eines Speichervorgangs in der oder den aktivierten Speichereinheiten mit wahlfreiem Zugriff der von den ersten Adressbits bestimmten Adresse entsprechend ermoeeglicht ist,
- ein erstes Datenuebertragungssystem (40, 41), das zum Empfang des Satzes von den

ersten Steuersignalen verbunden ist und die augenblicklich von dem logischen Schaltkreis aktivierte Speichereinheit mit wahlfreiem Zugriff an dem N-Byte-Datensammelbus zur Uebertragung eines N-Byte-Datensegments zwischen dem N-Byte-Datensammelbus und der aktivierten Speichereinheit mit wahlfreiem Zugriff koppelt,

- ein zweites Datenuebertragungssystem (44, 45), das zum Empfang des weiteren Steuersignals verbunden ist und im Ansprechen darauf gleichzeitig alle Speichereinheiten mit wahlfreiem Zugriff an dem M-Byte-Sammelbus zur parallelen Uebertragung eines M-Byte-Datensegments zwischen dem M-Byte-Sammelbus und den Speichereinheiten mit wahlfreiem Zugriff koppelt,
- Erzeugungsmittel von Steuersignalen zur Erzeugung des Satzes von ersten Steuersignalen und des weiteren Steuersignals, wobei die ersten Steuersignale in einer solchen Reihe erzeugt sind, dass Zugriff der Speichereinheiten mit wahlfreiem Zugriff von dem logischen Schaltkreis eines auf einmal in einer kreisende Reihenfolge erfolgt,
- wo das Daten-Schnittstellen-Mechanismus dadurch gekennzeichnet, dass:
- die ersten Adressbits die wertniedrigsten Adressbits fuer eine Adresse des Adressensammelbus des Mikroprozessors (38) bilden,
- die Steuersignale-Erzeugungsmittel einen Schaltkreis zur Code-Umsetzung (68) aufweist, der zum Empfang der werthoeheren Adressbits fuer eine Adresse des Adressensammelbus des Mikroprozessors verbunden ist, und der im Ansprechen auf den werthoeheren Adressbits fuer eine Adresse eines Adressbereichs aus einer Gruppe von ersten Adressbereichen ein entsprechendes Steuersignal aus den ersten Steuersignalen und im Ansprechen auf die werthoeheren Adressbits fuer eine Adresse in einem weiteren Adressbereich das weitere Steuersignal erzeugt, und
- der Mikroprozessor mit Mitteln versehen ist, die dem Sammelbus des Mikroprozessors eine Folge von ersten Adressen zuweisen, wobei jede Adresse zum einem aus der Gruppe der ersten Adressbereiche gehoert, so dass die Uebertragung von Daten mittels N-Byte-Segmente zwischen den Speichereinheiten mit wahlfreiem Zugriff und dem N-Byte-Datensammelbus veranlasst ist, und dem Adressbus des Mikroprozessors eine Adresse des weiteren Adressbereichs zur Veranlassung der parallelen Uebertragung von M-Datenbytes zwischen den Speichereinheiten mit wahlfreiem Zugriff und dem M-Byte Datenbus zuweisen.

2. Daten-Schnittstellen-Mechanismus nach Anspruch 1, in dem $M=2$ und $N=1$, wo zwei

Speichereinheiten mit wahlfreiem Zugriff zwischen dem Zwei-Byte-Bus (34) und dem Ein-Byte-Bus (37) angeordnet sind, wobei jede Speichereinheit eine Breite von einem Byte aufweist.

3. Daten-Schnittstellen-Mechanismus nach Anspruch 1 oder 2, dadurch gekennzeichnet, dass der Mechanismus eine Steuereinheit (13) mit direktem Speicherzugriff zur Lieferung einiger Adressen aufweist, die fuer den Zugriff zu den Speichereinheiten (22) angewendet sind.

Revendications

1. Mécanisme d'interface de données pour assurer l'interface d'un bus de données à M multiplets (8, 34) connecté à un processeur principal (1) avec un bus de données à N multiplets (16, 37) dans lequel le rapport de M à N est un nombre entier plus grand que 1, lequel bus de données à N multiplets est connecté à des dispositifs E/S (3—6) par l'intermédiaire d'un microprocesseur (11), comprenant:

un certain nombre d'unités d'emmaganage à accès aléatoire (22) qui est égal au rapport de M à N, chacune de ces unités d'emmaganage ayant une largeur de N multiplets,

un circuit d'adressage commun (63) connecté au bus d'adresses (38) du microprocesseur pour recevoir des premiers bits d'adresse et pour délivrer lesdits premiers bits d'adresse à toutes les unités d'emmaganage à accès aléatoire,

un circuit logique (48) qui est connecté pour recevoir un ensemble de premiers signaux de commande (CS1, CS2) et un autre signal de commande (CS3) pour activer une unité différente desdites unités d'emmaganage à accès aléatoire en réponse à chacun desdits premiers signaux de commande, et pour activer simultanément toutes lesdites unités d'emmaganage à accès aléatoire en réponse audit autre signal de commande, ce qui permet l'exécution d'une opération d'emmaganage dans l'unité ou les unités d'emmaganage à accès aléatoire activées à l'adresse spécifiée par lesdits premiers bits d'adresse,

un premier mécanisme de transfert de données (40, 41) qui est connecté pour recevoir ledit ensemble de premiers signaux de commande et qui, en réponse à chacun desdits premiers signaux de commande, raccorde l'unité d'emmaganage à accès aléatoire actuellement activée par le circuit logique, au bus de données à N multiplets pour transférer un segment de donnée à N multiplets entre le bus de données à N multiplets et ladite unité d'emmaganage à accès aléatoire activée, un deuxième mécanisme de transfert de

données (44, 45) qui est connecté pour recevoir ledit autre signal de commande et qui, en réponse à celui ci, raccorde simultanément toutes les unités d'emmaganage à accès aléatoire au bus de données à M multiplets pour transférer en parallèle un segment de données à M multiplets entre ledit bus de données à M multiplets et lesdites unités d'emmaganage à accès aléatoire, et

un moyen de génération de signaux de commande pour générer ledit ensemble de premiers signaux de commande et ledit autre signal de commande, lesdits premiers signaux de commande étant générés dans un ordre tel que lesdites unités d'emmaganage à accès aléatoire sont accédées une par une, d'une manière rotative, par ledit circuit logique,

ledit mécanisme d'interface de données étant caractérisé en ce que:

lesdits premiers bits d'adresse forment sur les bits d'adresse d'ordre d'inférieur d'une adresse sur le bus d'adresses de microprocesseur (38),

ledit moyen de génération de signaux de commande comprend un circuit décodeur (68) qui est connecté pour recevoir les bits d'adresse d'ordre plus élevé d'une adresse sur le bus d'adresses du microprocesseur et qui, en réponse aux bits d'adresse d'ordre plus élevé d'une adresse dans l'un d'un ensemble de premières plages d'adresses, génère un signal respectif desdits premiers signaux de commande et, en réponse aux bits d'adresse d'ordre plus élevé, d'une adresse dans une autre plage d'adresses, génère le dit autre signal de commande, et en ce que

ledit microprocesseur comporte des moyens pour placer sur ledit bus d'adresses de microprocesseur, une série de premières adresses, chacune d'elles étant une adresse dans l'une desdites plages d'adresses, afin de provoquer le transfert de données par des segments à N multiplets entre les éléments d'emmaganage à accès aléatoire et le bus de données à N multiplets, et pour placer sur ledit bus d'adresses de microprocesseur, une adresse dans ladite autre plage d'adresser afin de provoquer le transfert en parallèle de M multiplets de données entre les unités d'emmaganage à accès aléatoire et le bus de données à M multiplets.

2. Mécanisme d'interface de données selon la revendication 1 dans lequel $M=2$ et $N=1$, deux unités d'emmaganage à accès aléatoire étant disposées entre le bus à deux multiplets (34) et le bus à un multiplet (37), chacune de ces unités d'emmaganage ayant une largeur de un multiplet.

3. Mécanisme d'interface de données selon la revendication 1 ou 2 caractérisé en ce qu'il

comprend en outre une unité de commande
d'accès en mémoire direct (13) pour délivrer

certaines des adresses utilisées pour accéder
auxdites unités d'emmagasinement (22).

5

10

15

20

25

30

35

40

45

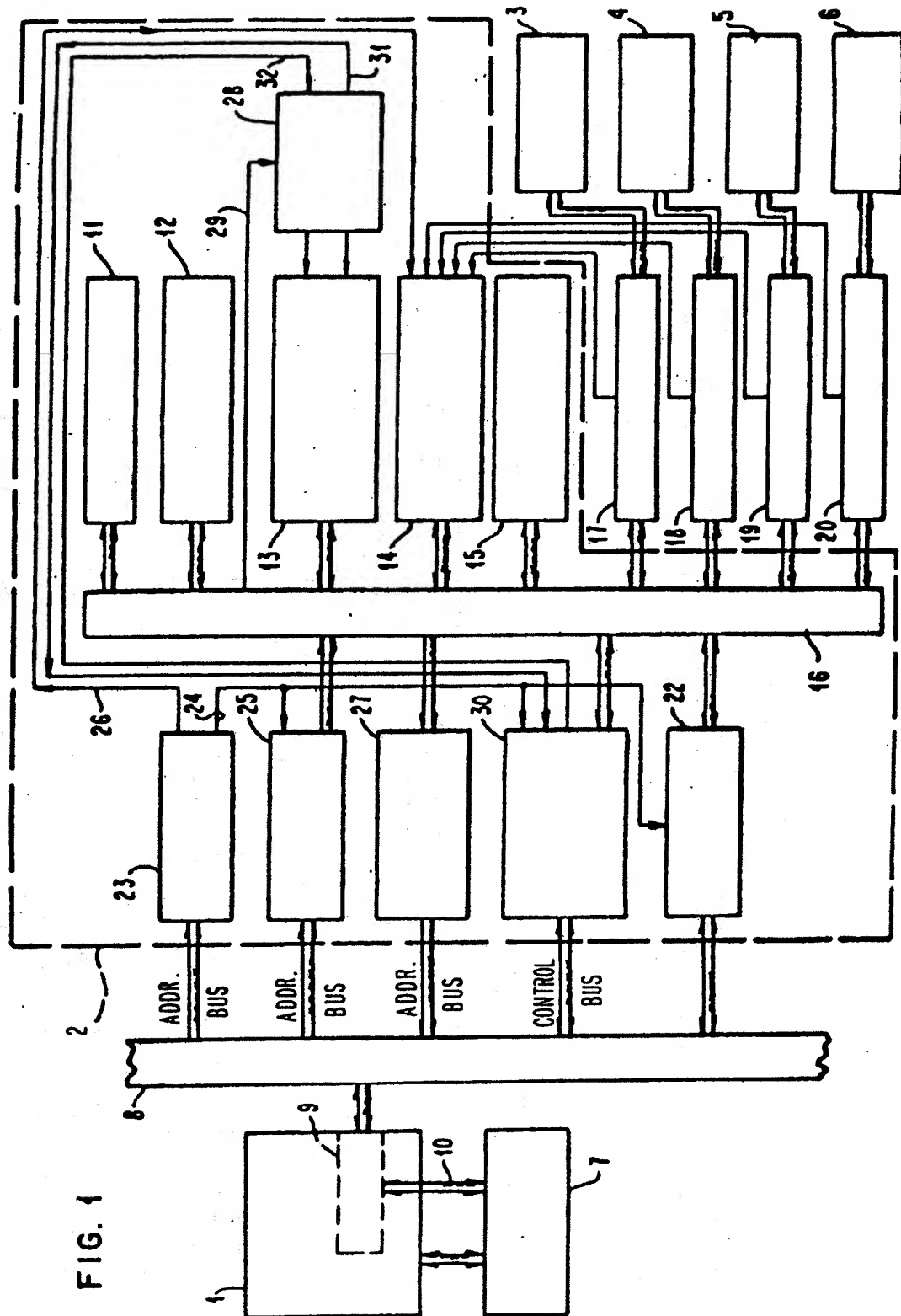
50

55

60

65

26



0 023 568

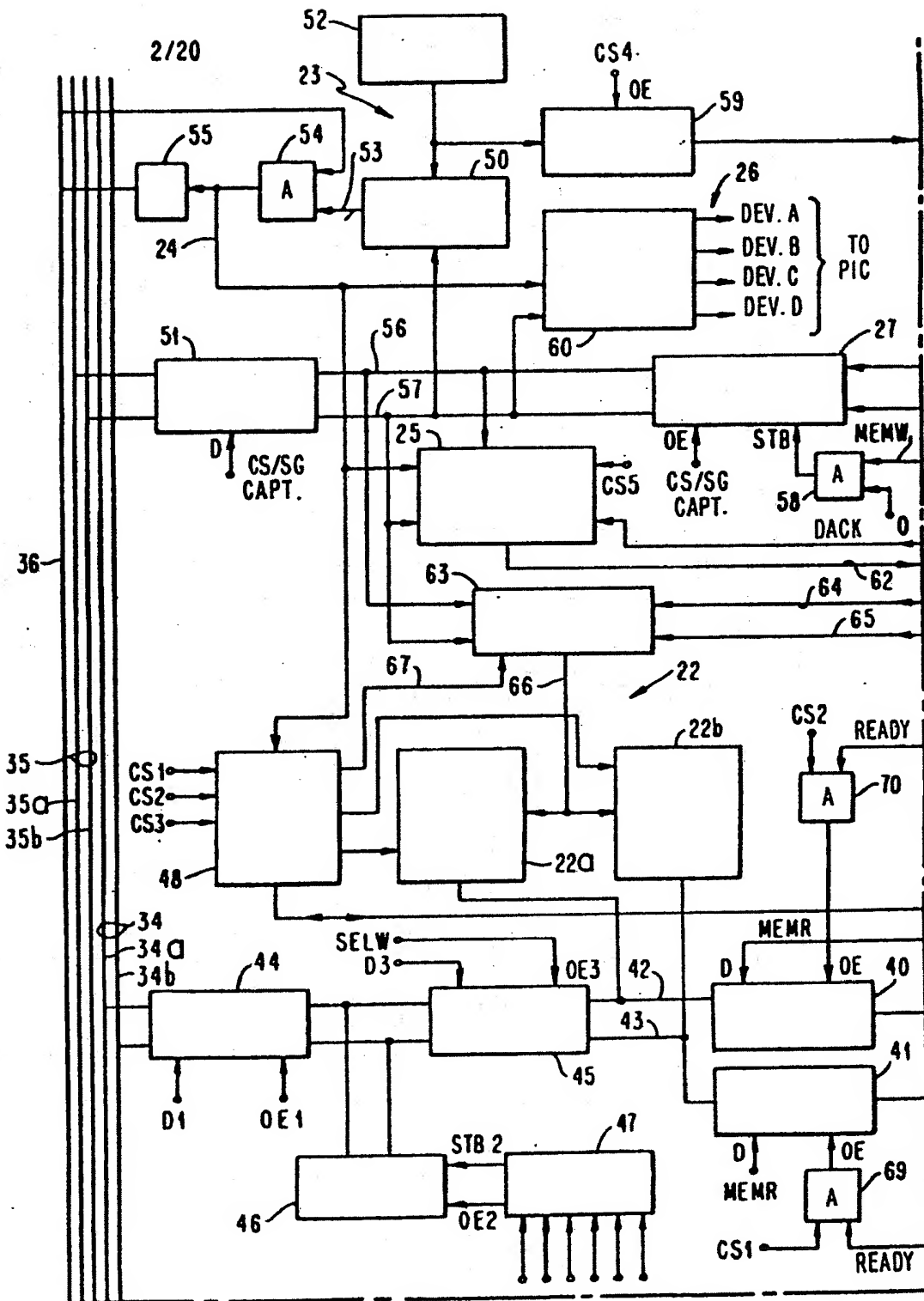
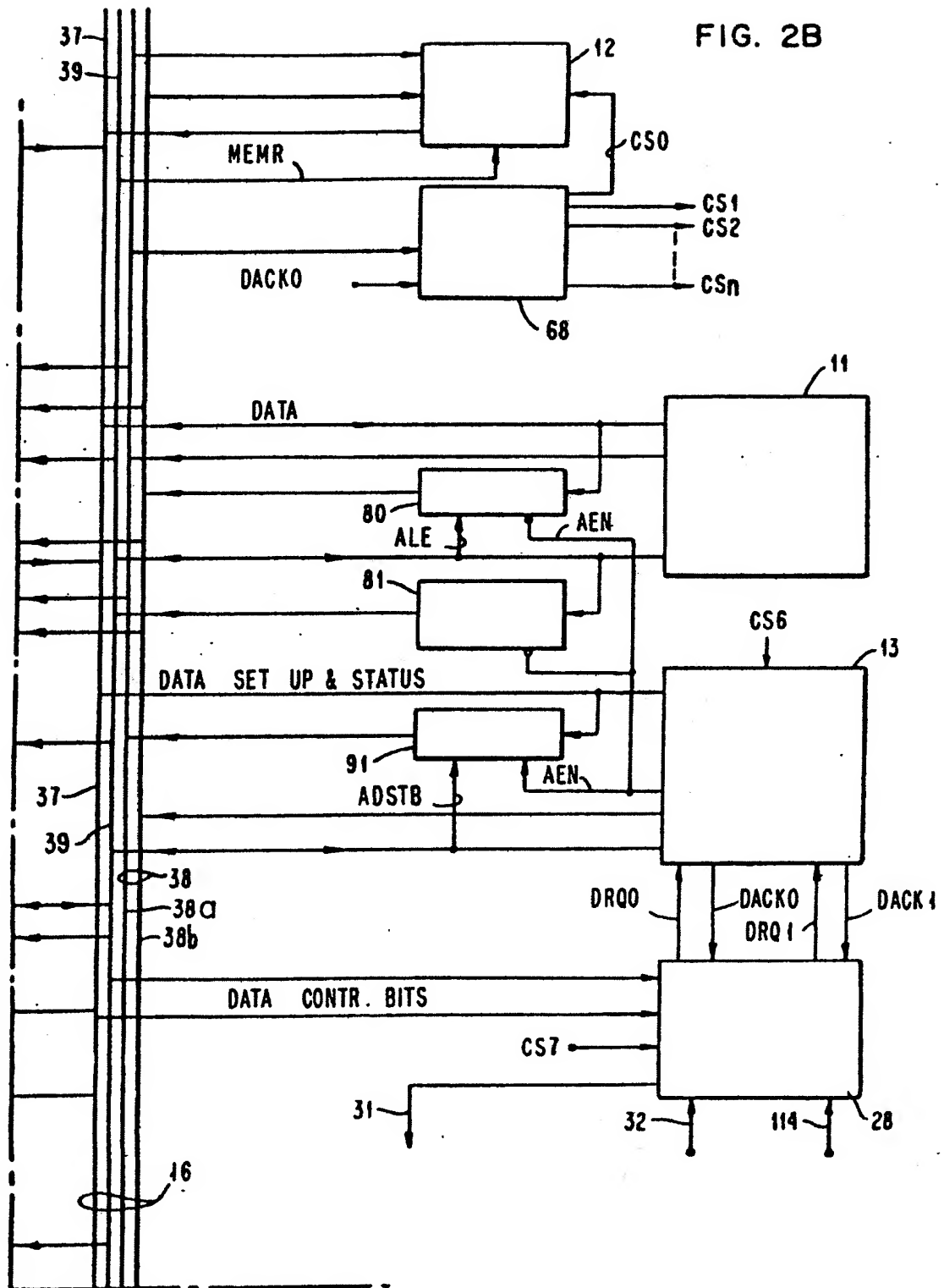


FIG. 2B



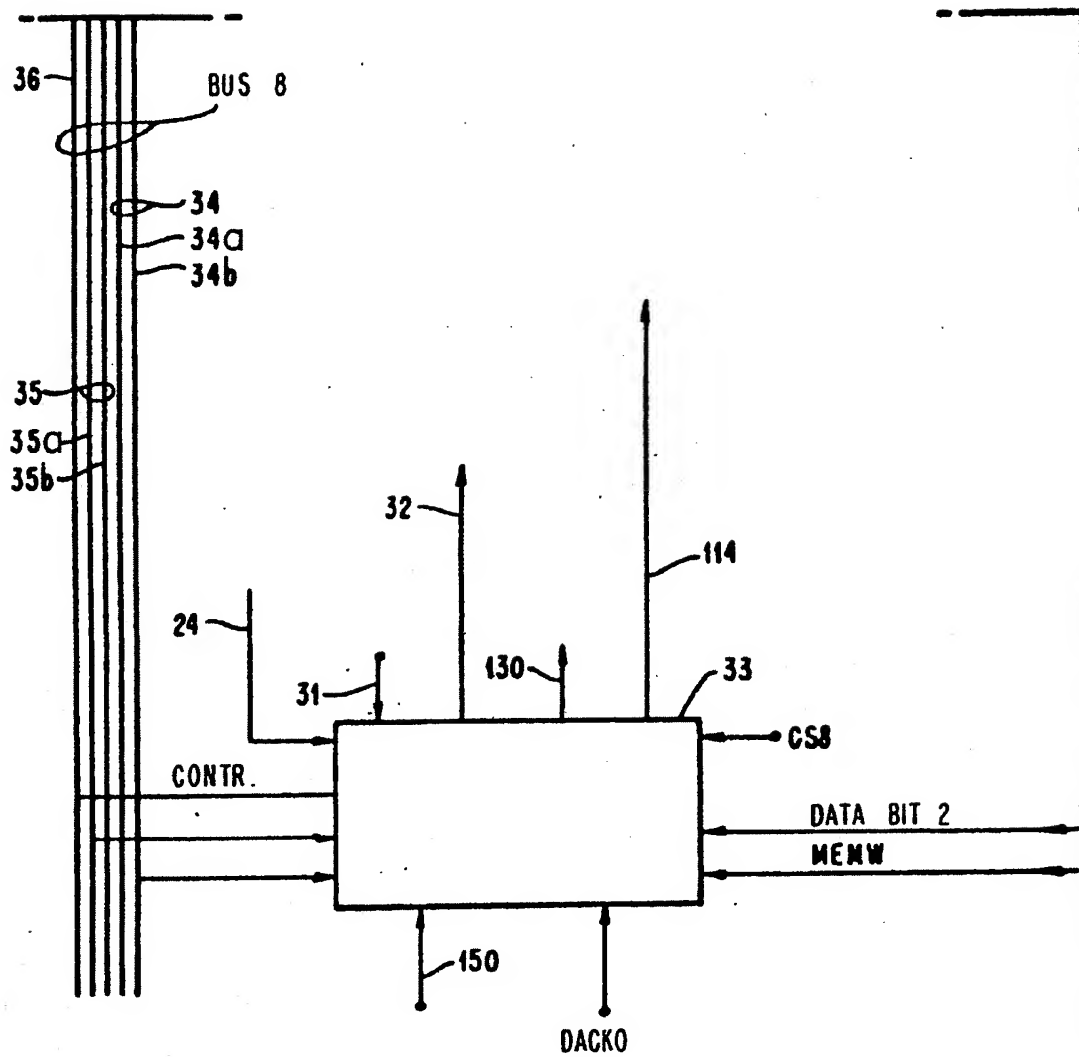


FIG. 2C

FIG. 2A	FIG. 2B
FIG. 2C	FIG. 2D

FIG. 2

FIG. 2D

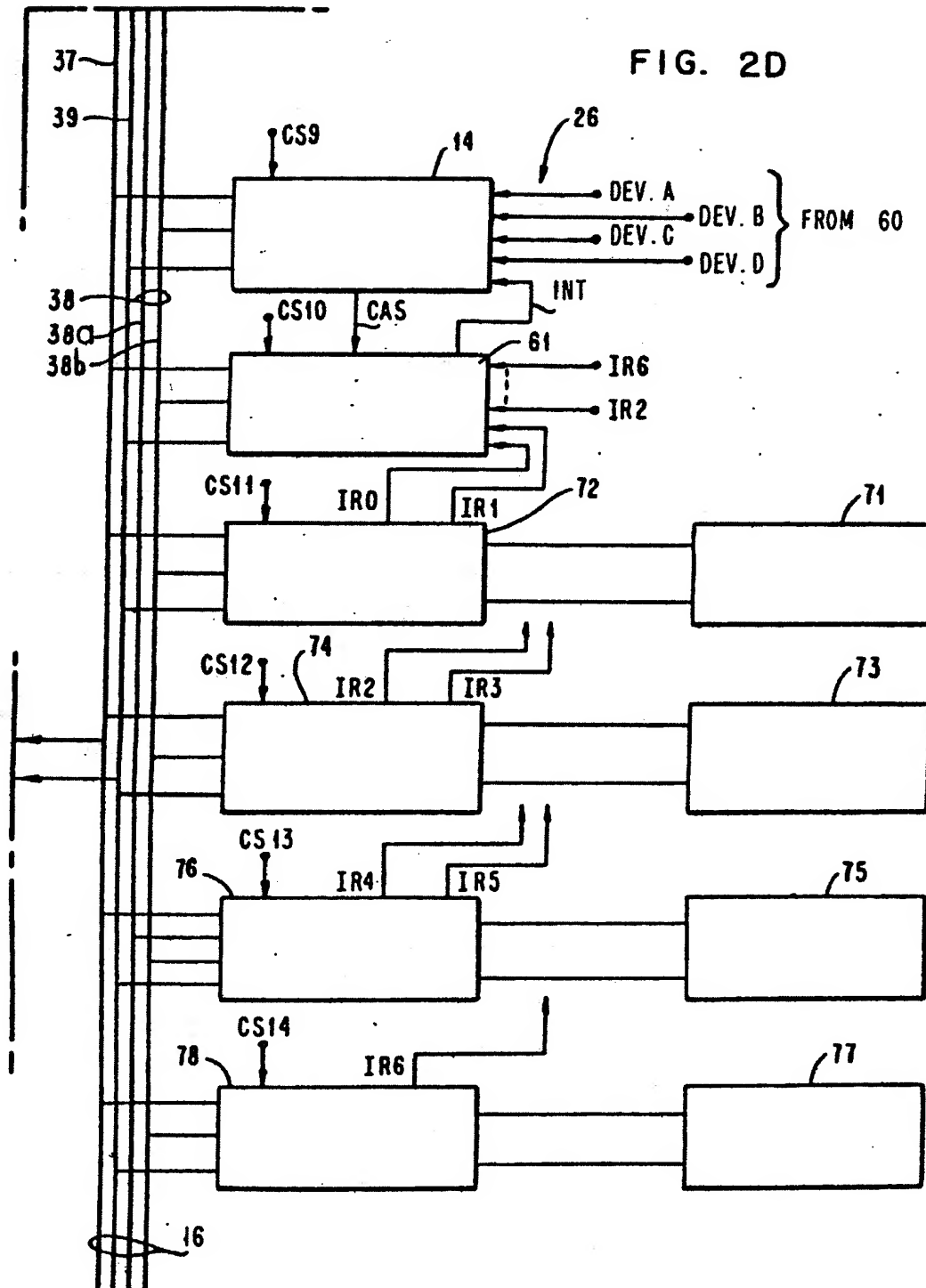


FIG. 3

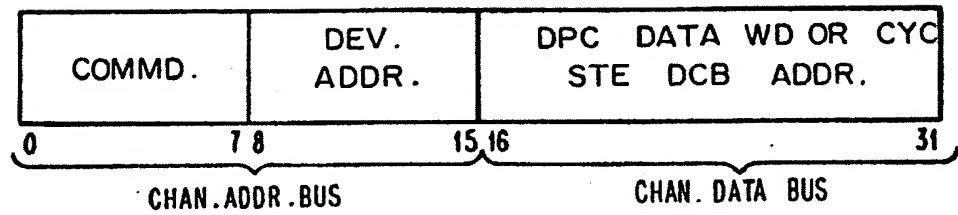


FIG. 4

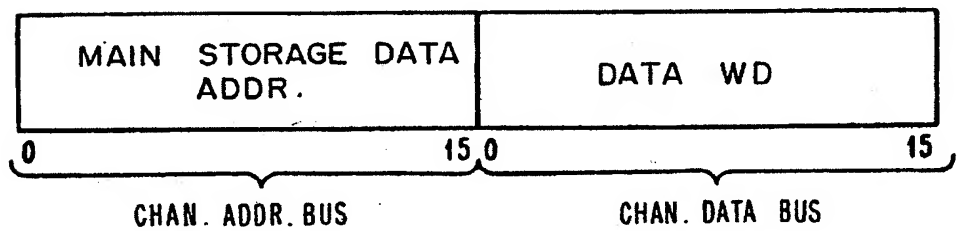


FIG. 5

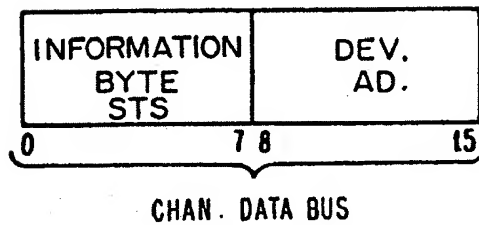


FIG. 6

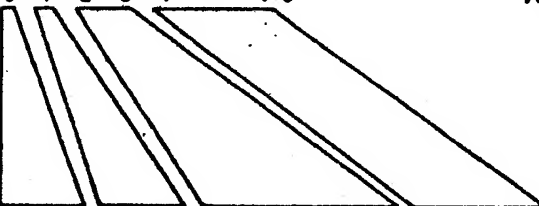
COMMD					DEV ADR		IMMEDIATE				FIELD	
0	1	2	3	4	7	8	15, 16		23, 24		31	
												
CHAN	R/W	FUNCTION		MODIFIER		HEX	SPECIFIC COMMAND		TYPE OF OPERATION			
0	0	00	RD	X	X	X	X	0X	RD 0	DPC		
0	0	01	RD	X	X	X	X	1X	RD 1	DPC		
0	0	10	RD STS	X	X	X	X	2X	RD STS	DPC		
0	0	11						3X				
0	1	00	WR	X	X	X	X	4X	WR 0	DPC		
0	1	01	WR	X	X	X	X	5X	WR 1	DPC		
0	1	10	CONTR	X	X	X	X	6X	CONTR	DPC		
0	1	11	ST	X	X	X	X	7X	ST CYC STE	CYC STE		
0	1	11	ST	1	1	1	1	7F	ST CYC STE STS	CYC STE		
1	1	11	CHAN	0	0	0	0	F0	H I/O	CHAN		

FIG. 7

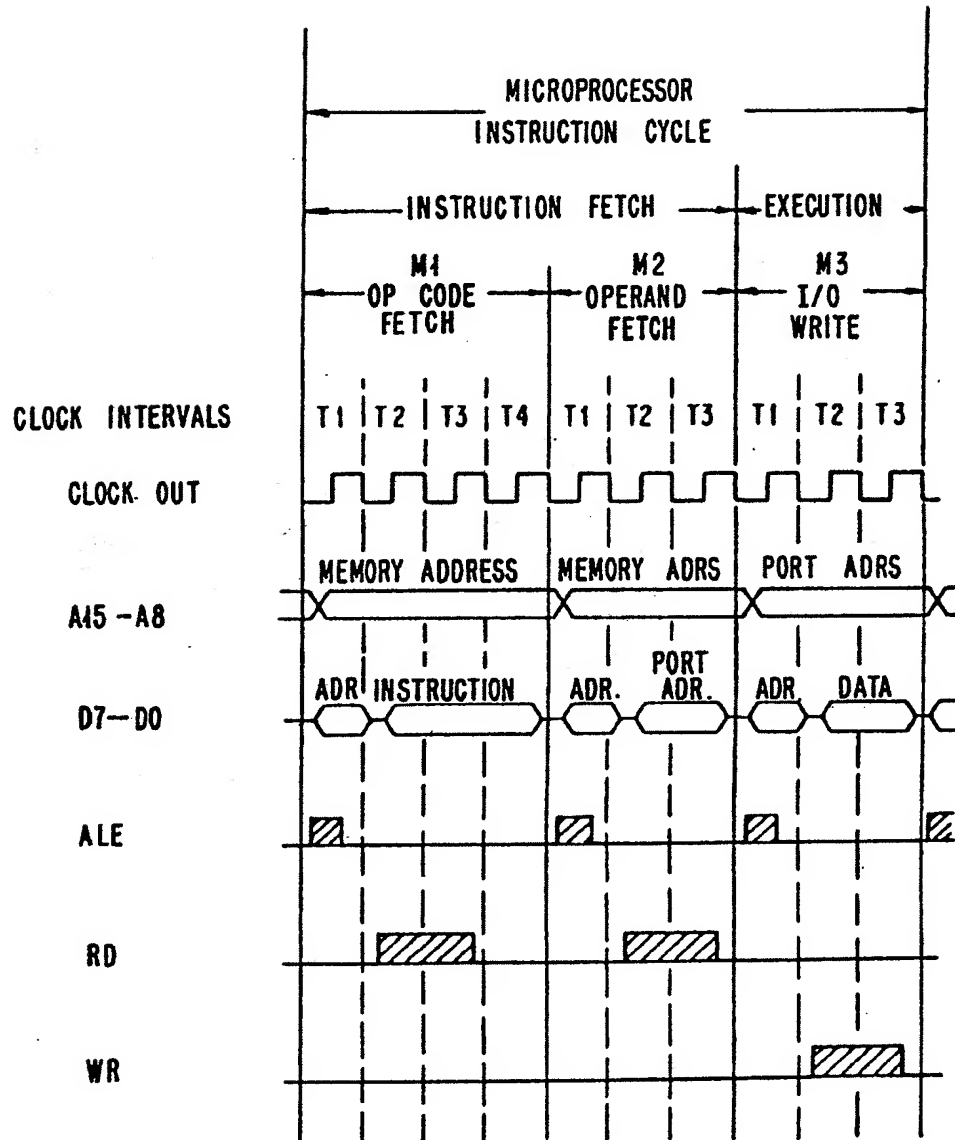


FIG. 8

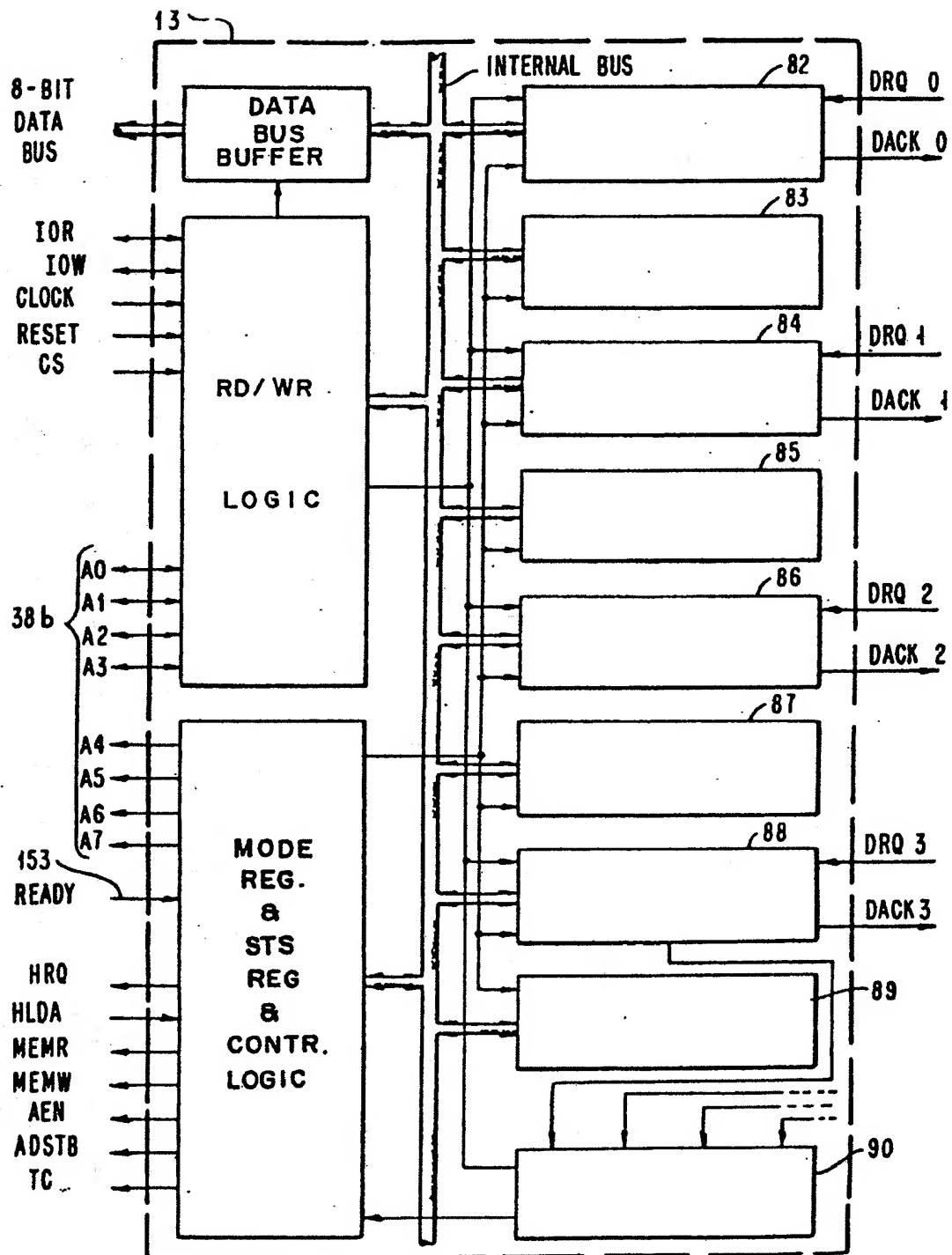


FIG. 9

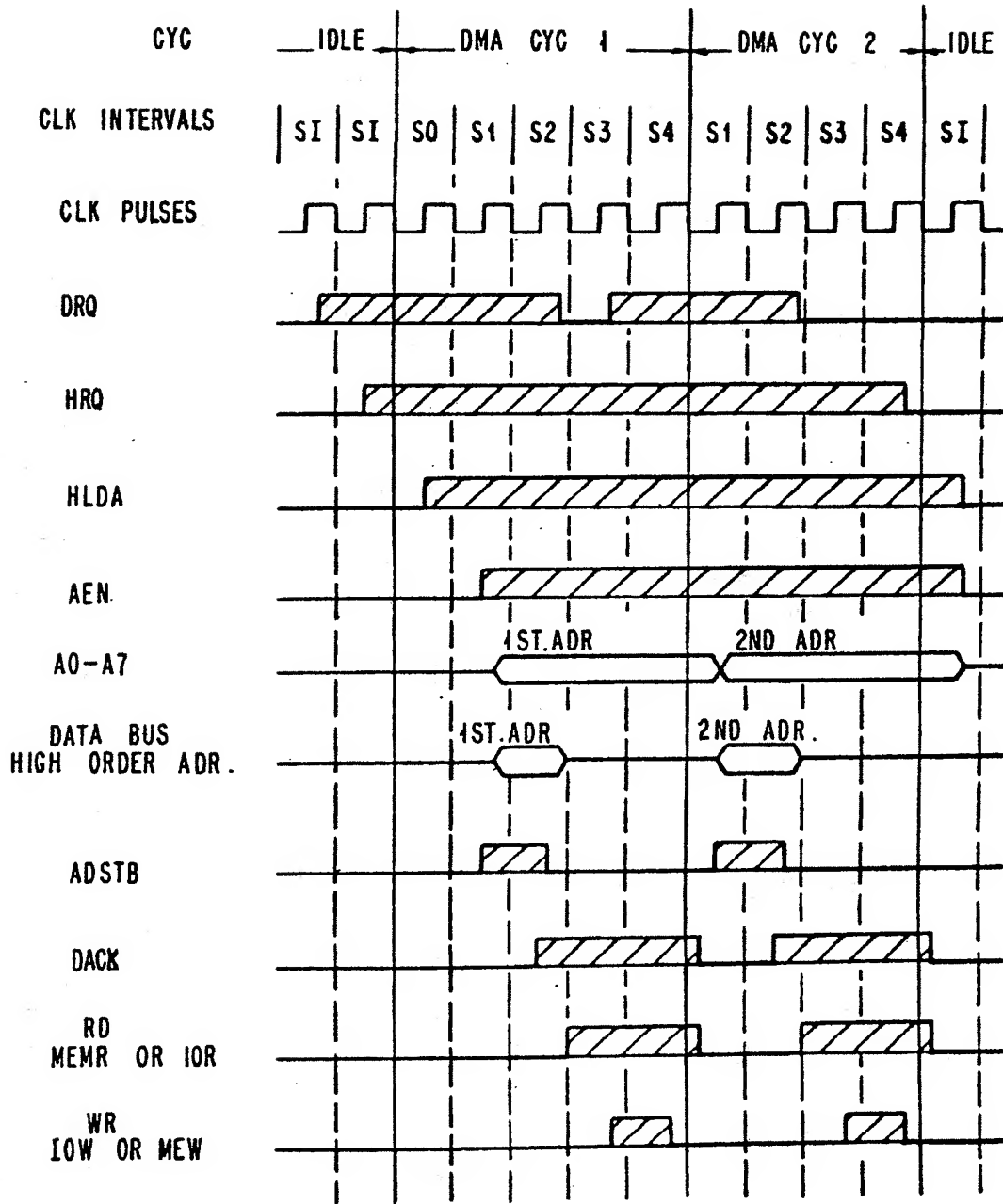


FIG. 10A

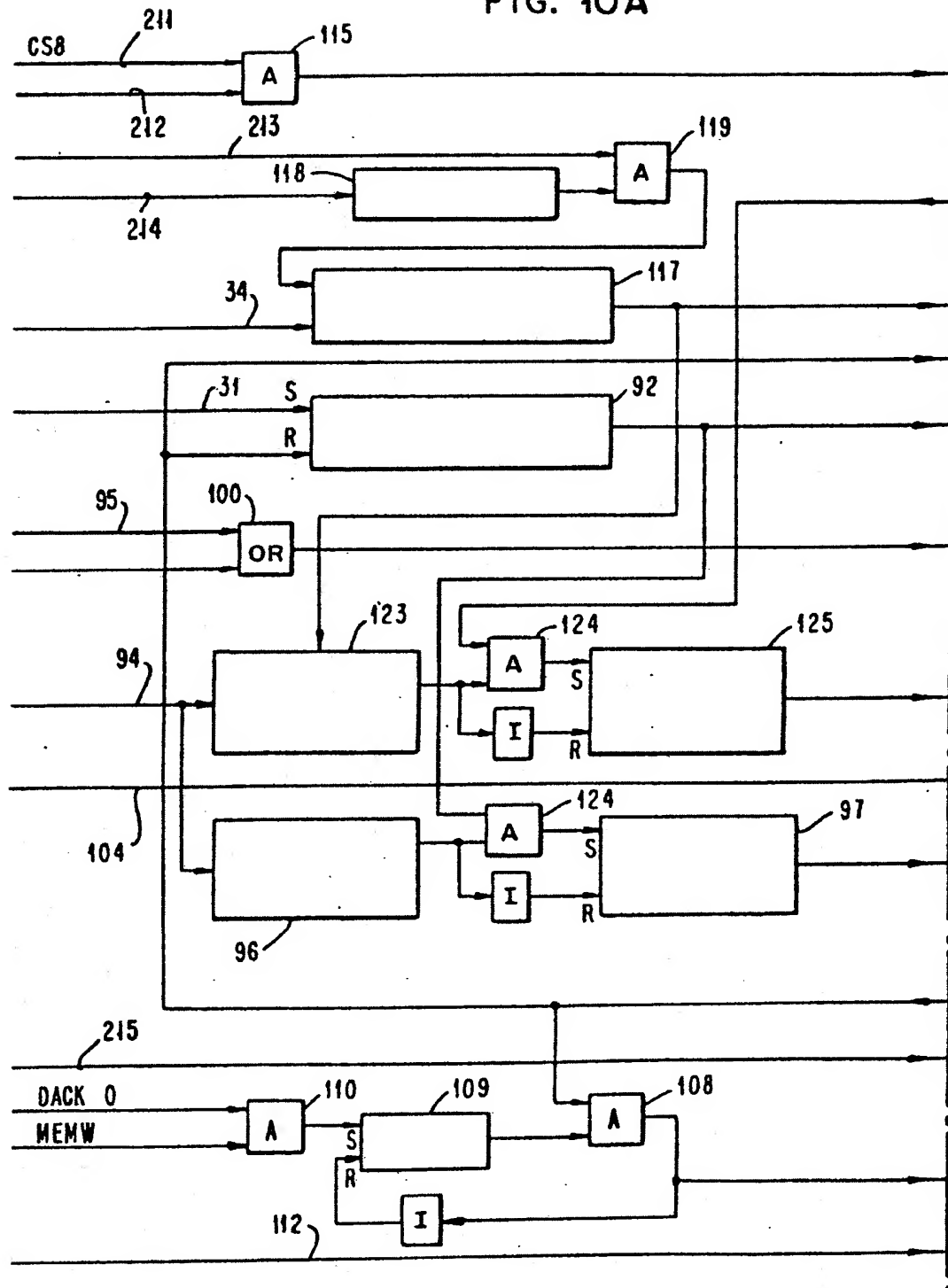


FIG. 10

FIG. 10A	FIG. 10B
-------------	-------------

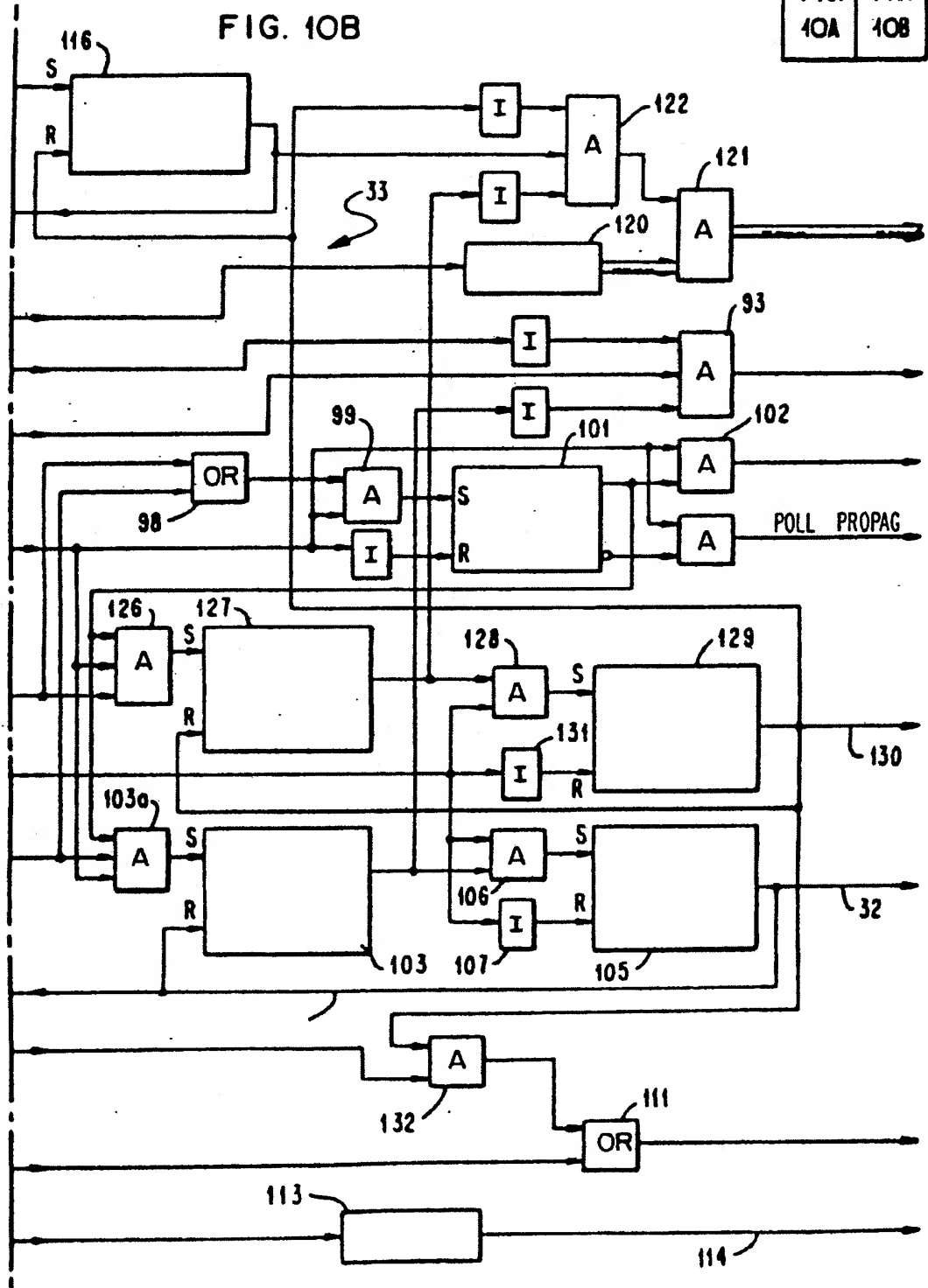
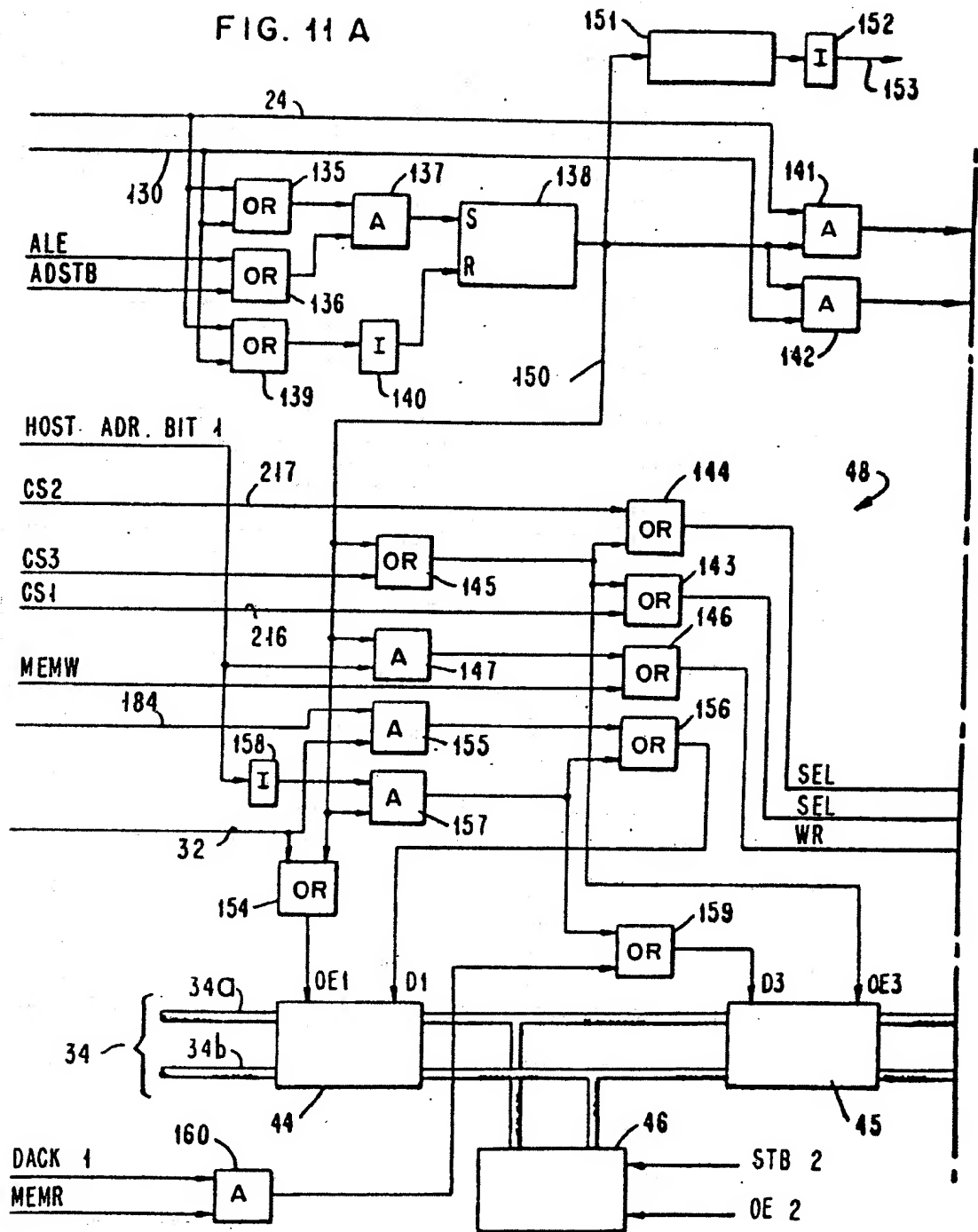


FIG. 11 A



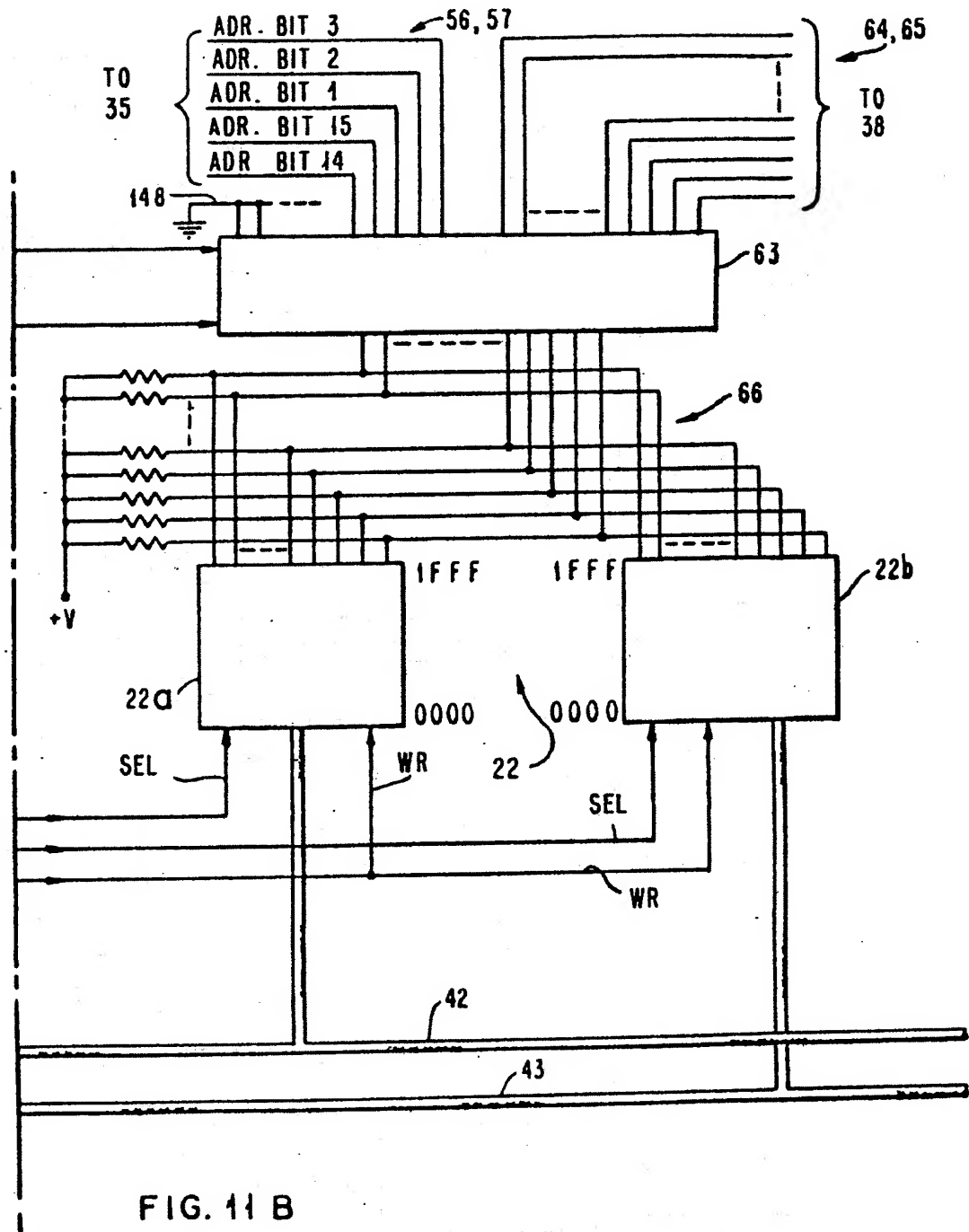


FIG. 11A	FIG. 11B
-------------	-------------

FIG. 11

FIG. 12

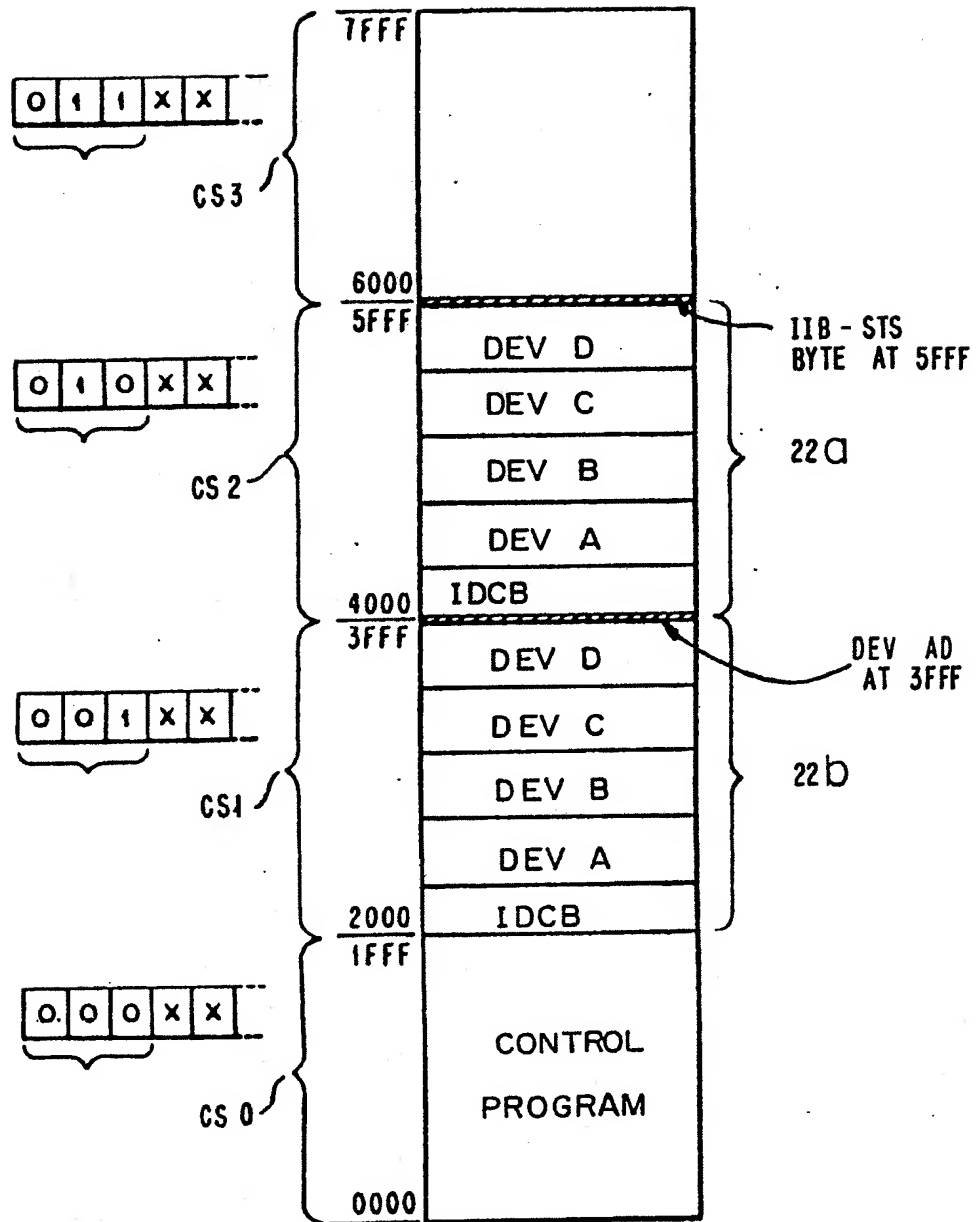


FIG. 13

HOST					STORAGE MAP FOR				
AD. BITS					IDCB BYTES 2 & 3				
14	15	1	2	3					
1	1	1	1	1	DEV	D	ST	31	
1	1	1	1	0	DEV	D	CONTR		
1	1	1	0	1	DEV	D	WR 1		
1	1	1	0	0	DEV	D	WR 0		
1	1	0	1	1					
1	1	0	1	0	DEV	D	RD STS		
1	1	0	0	1	DEV	D	RD 1		
1	1	0	0	0	DEV	D	RD 0		
1	0	1	1	1	DEV	C	ST		
1	0	1	1	0	DEV	C	CONTR		
1	0	1	0	1	DEV	C	WR 1		
1	0	1	0	0	DEV	C	WR 0		
1	0	0	1	1					
1	0	0	1	0	DEV	C	RD STS		
1	0	0	0	1	DEV	C	RD 1		
1	0	0	0	0	DEV	C	RD 0		
0	1	1	1	1	DEV	B	ST		
0	1	1	1	0	DEV	B	CONTR		
0	1	1	0	1	DEV	B	WR 1		
0	1	1	0	0	DEV	B	WR 0		
0	1	0	1	1					
0	1	0	1	0	DEV	B	RD STS		
0	1	0	0	1	DEV	B	RD 1		
0	1	0	0	0	DEV	B	RD 0		
0	0	1	1	1	DEV	A	ST		
0	0	1	1	0	DEV	A	CONTR		
0	0	1	0	1	DEV	A	WR 1		
0	0	1	0	0	DEV	A	WR 0		
0	0	0	1	1					
0	0	0	1	0	DEV	A	RD STS		
0	0	0	0	1	DEV	A	RD 1		
0	0	0	0	0	DEV	A	RD		

DEV
AD

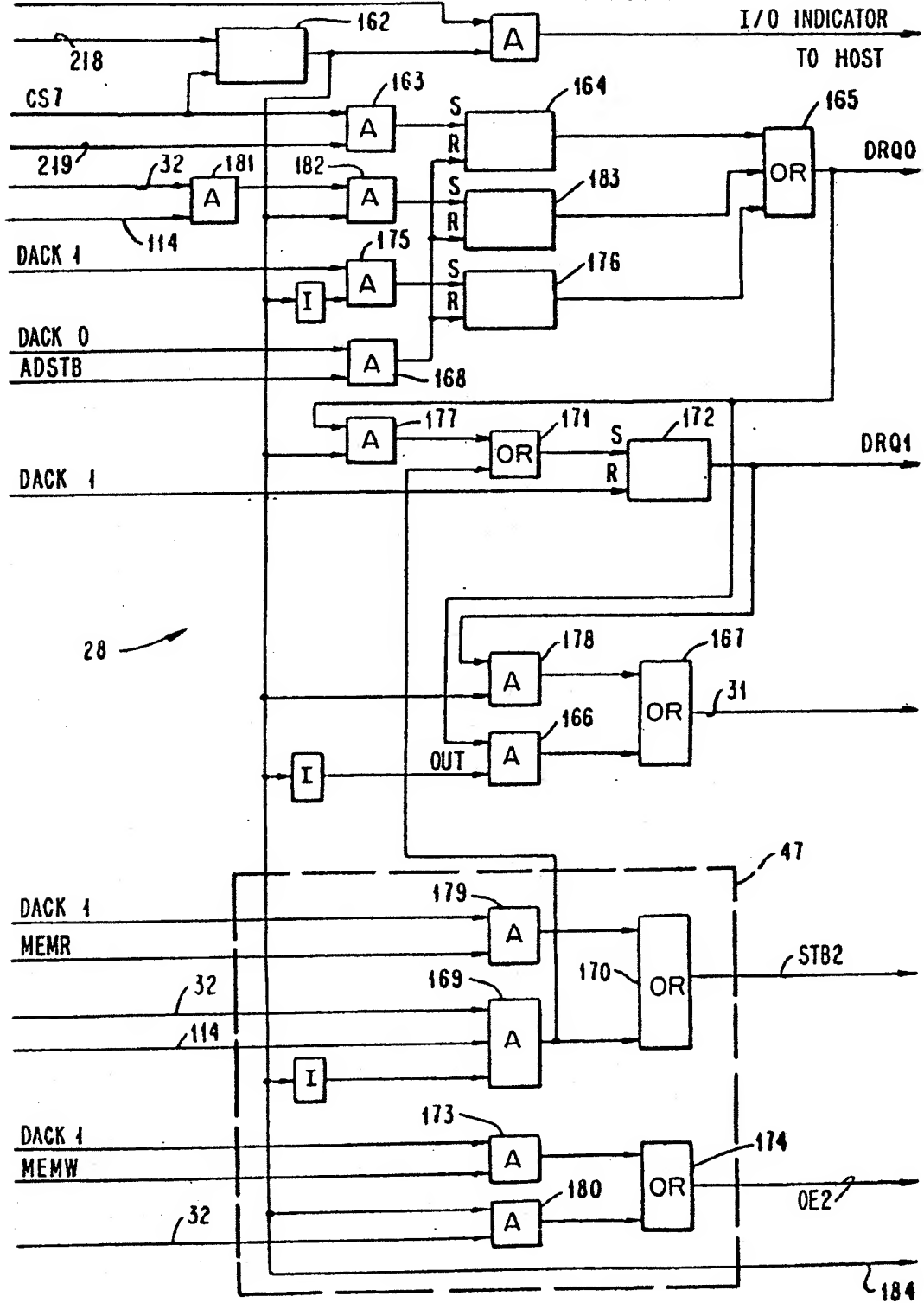
DEV
AD

COMMD
BITS

DEV AD BITS COMMD BITS

CS/SG CAPT.

FIG. 14



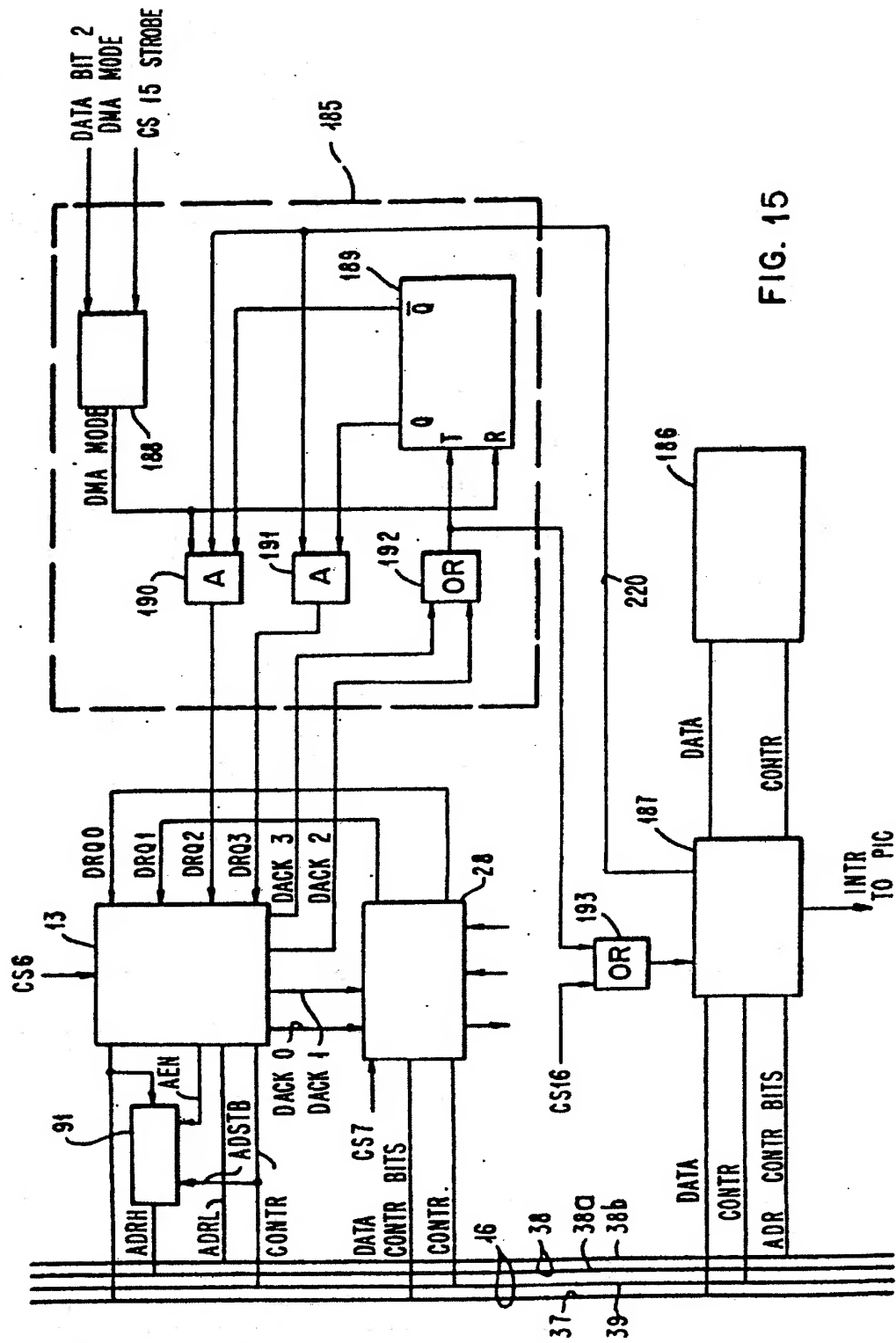


FIG. 15

FIG. 16

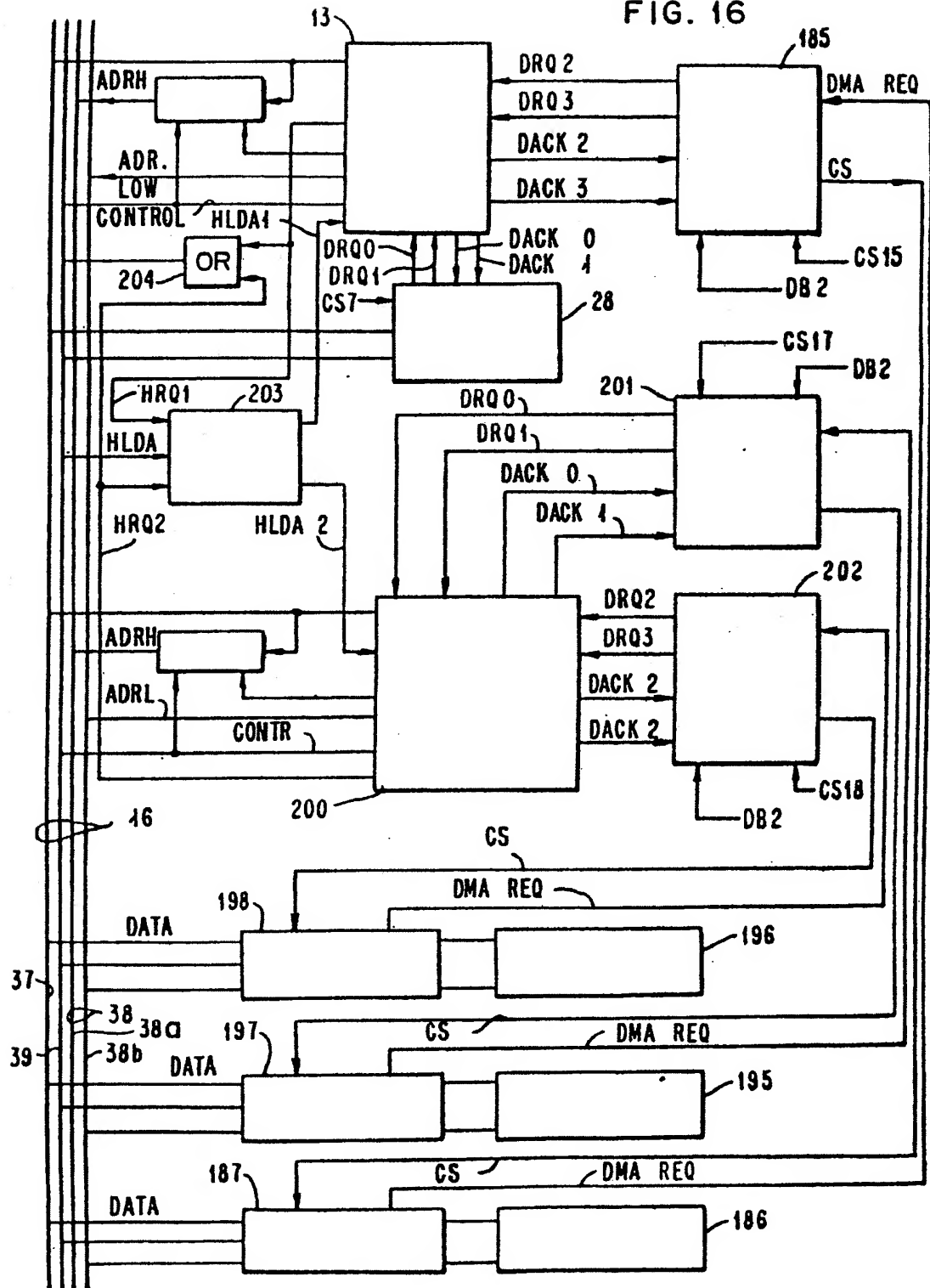


FIG. 17

